Flokzu README - Workflow Configuration Guidelines

This document provides formal guidelines for configuring and managing workflows in Flokzu. It emphasizes best practices, warnings, and procedural steps to ensure reliable process execution, particularly when handling versioning, field visibility, scripts, gateways, and integrations with databases or external forms. All references to images and examples are preserved for illustrative purposes.

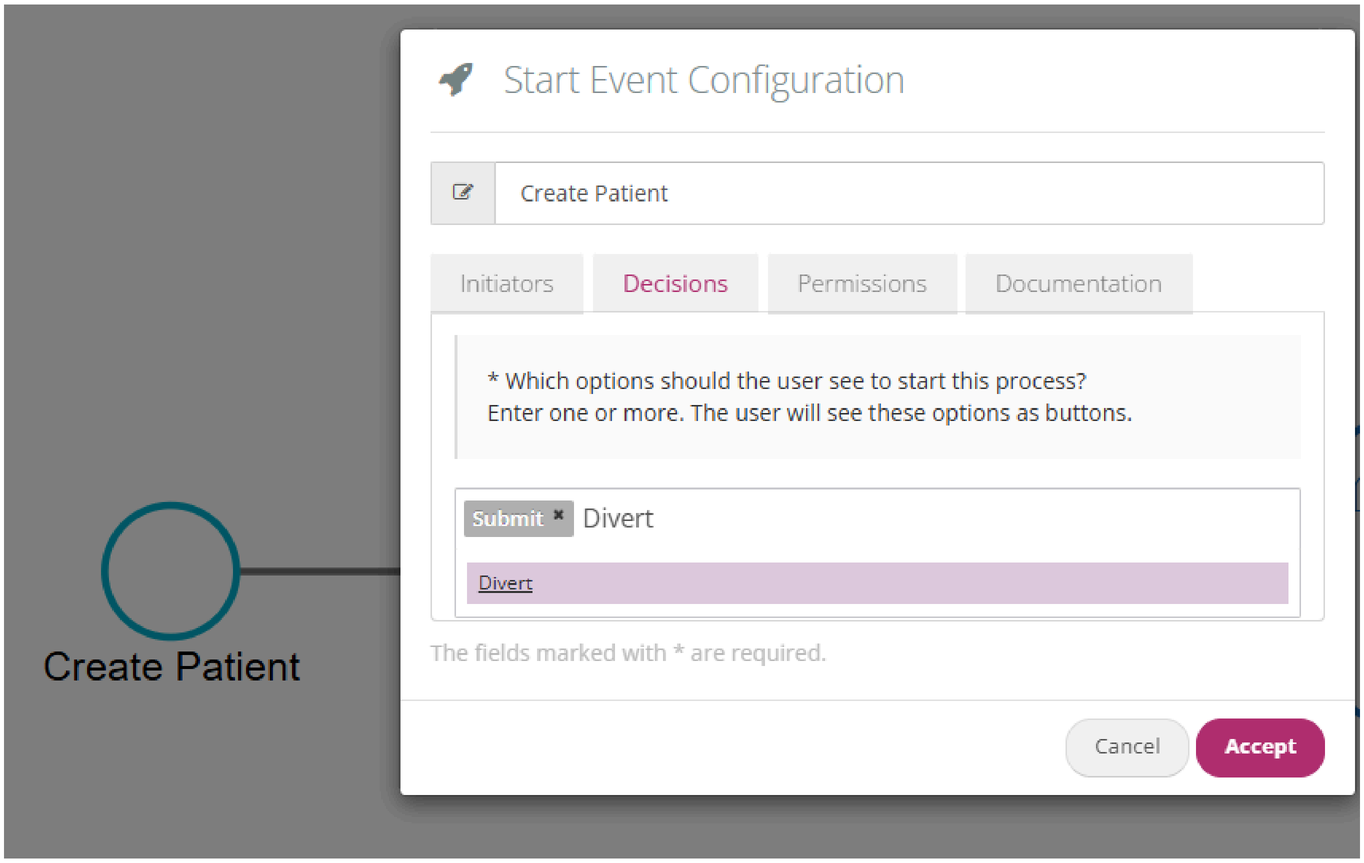
Permitted Modifications

Procedure for Implementing Decisions and Exclusive Gateways

Decisions and exclusive gateways are highly effective mechanisms for finalizing an existing process instance and re-initiating it under a revised process version, particularly in scenarios where direct migration is not feasible. This approach enables the selective bypassing of steps for which data is already available.

Step 1: Insert Decision and Exclusive Gateway at the Required Task

(The insertion is most commonly performed at the process launch task but may be applied to other tasks as necessary.)


Start Event Configuration

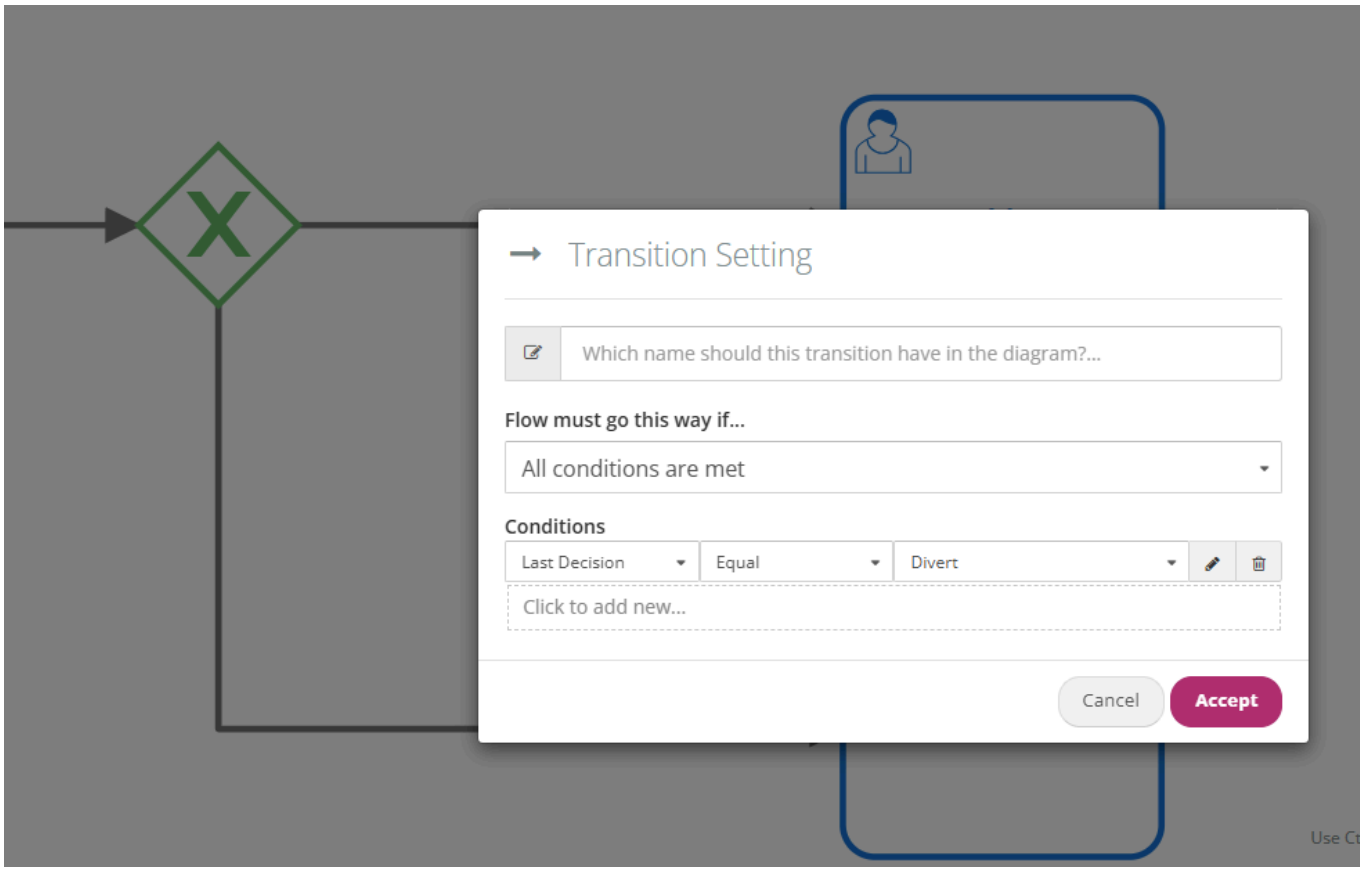
Create Patient

* Which options should the user see to start this process?
Enter one or more. The user will see these options as buttons.

Submit × Divert

Divert

The fields marked with * are required.


Transition Setting

Which name should this transition have in the diagram?...

Flow must go this way if...

All conditions are met

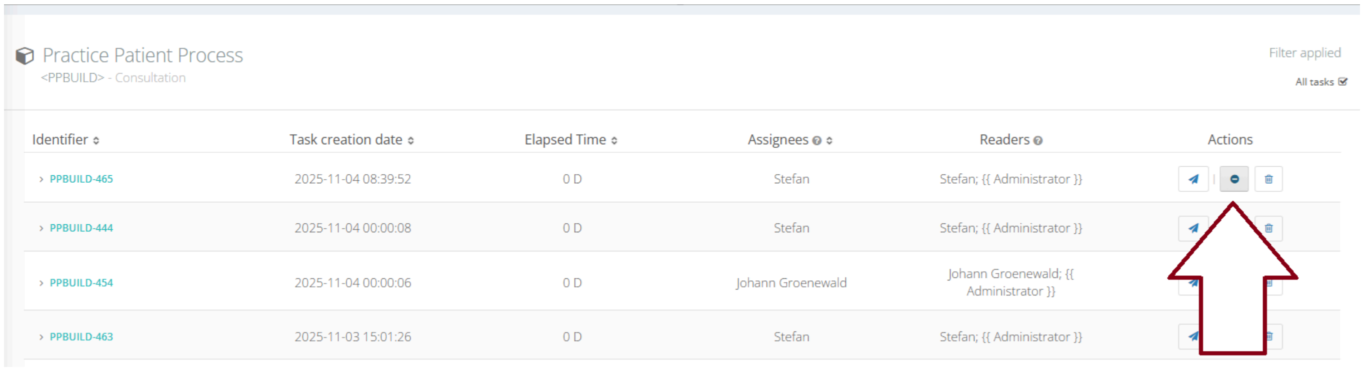
Conditions

Last Decision	Equal	Divert

Click to add new...

Step 2: Save and Deploy the Modified Process

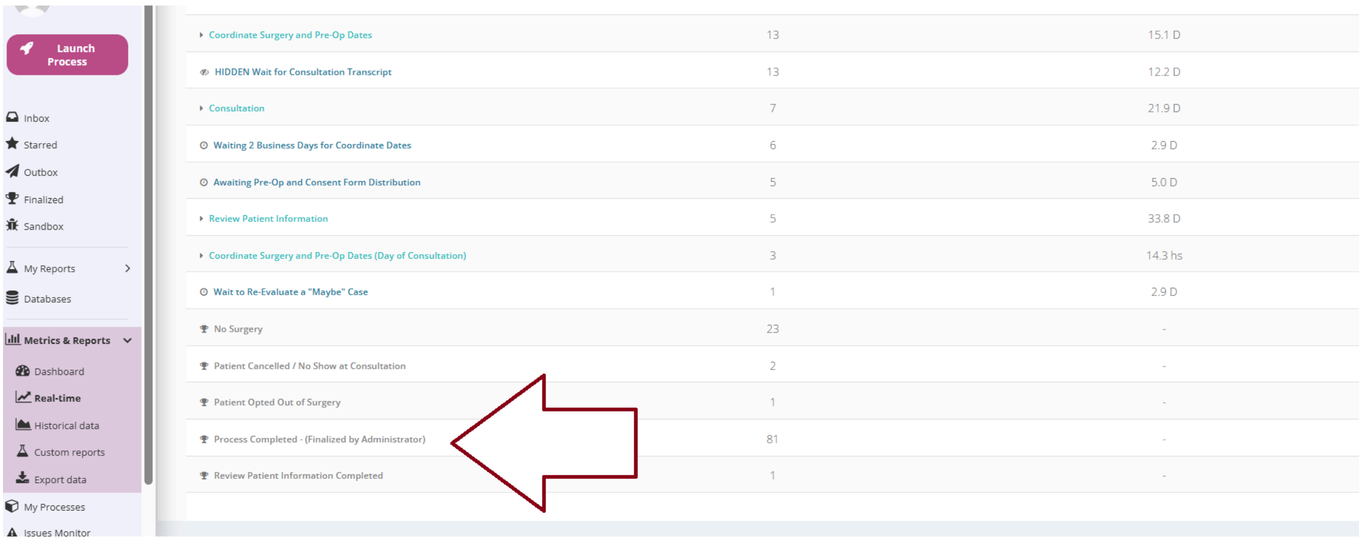
Step 3: Terminate the Existing Process Instance


Practice Patient Process
<PPBUILD> - Consultation

Filter applied
All tasks

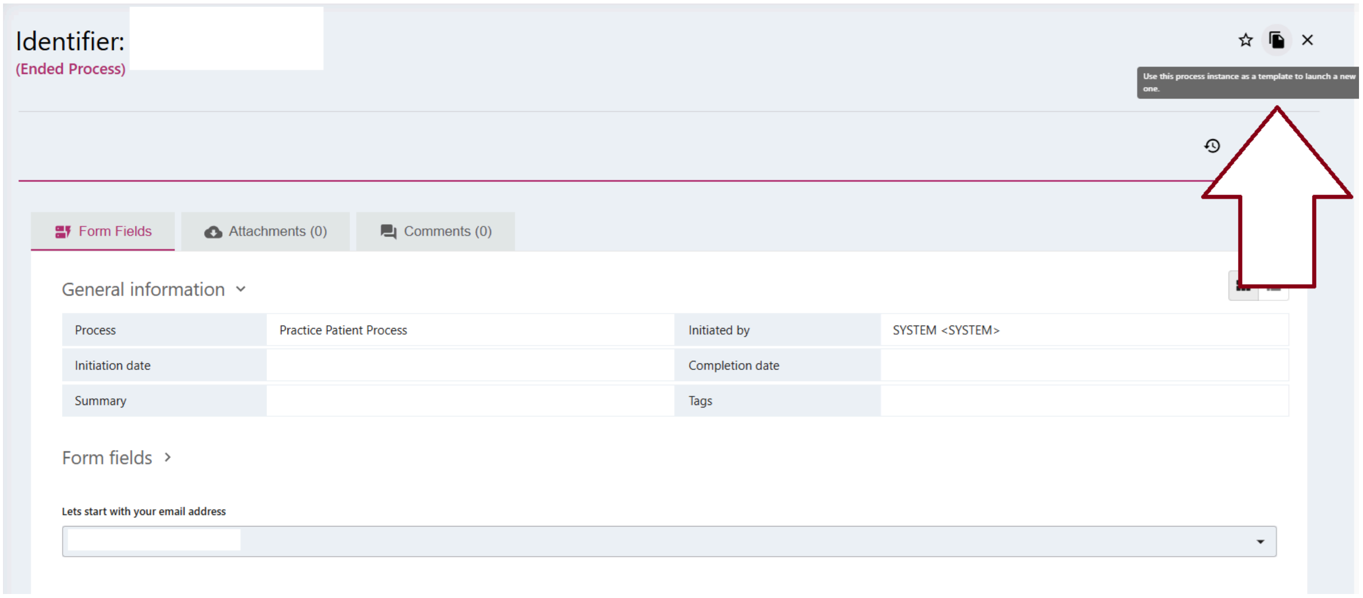
Identifier	Task creation date	Elapsed Time	Assignees	Readers	Actions
PPBUILD-465	2025-11-04 08:39:52	0 D	Stefan	Stefan; {{ Administrator }}	
PPBUILD-444	2025-11-04 00:00:08	0 D	Stefan	Stefan; {{ Administrator }}	
PPBUILD-454	2025-11-04 00:00:06	0 D	Johann Groenewald	Johann Groenewald; {{ Administrator }}	
PPBUILD-463	2025-11-03 15:01:26	0 D	Stefan	Stefan; {{ Administrator }}	

Step 4: Locate the Finalized Process Instance in Real Time



Coordinate Surgery and Pre-Op Dates	13	15.1 D
HIDDEN Wait for Consultation Transcript	13	12.2 D
Consultation	7	21.9 D
Waiting 2 Business Days for Coordinate Dates	6	2.9 D
Awaiting Pre-Op and Consent Form Distribution	5	5.0 D
Review Patient Information	5	33.8 D
Coordinate Surgery and Pre-Op Dates (Day of Consultation)	3	14.3 hs
Wait to Re-Evaluate a "Maybe" Case	1	2.9 D
No Surgery	23	-
Patient Cancelled / No Show at Consultation	2	-
Patient Opted Out of Surgery	1	-
Process Completed - (Finalized by Administrator)	81	-
Review Patient Information Completed	1	-

Step 5: Duplicate the Instance


Identifier:
(Ended Process)

Use this process instance as a template to launch a new one.

General information

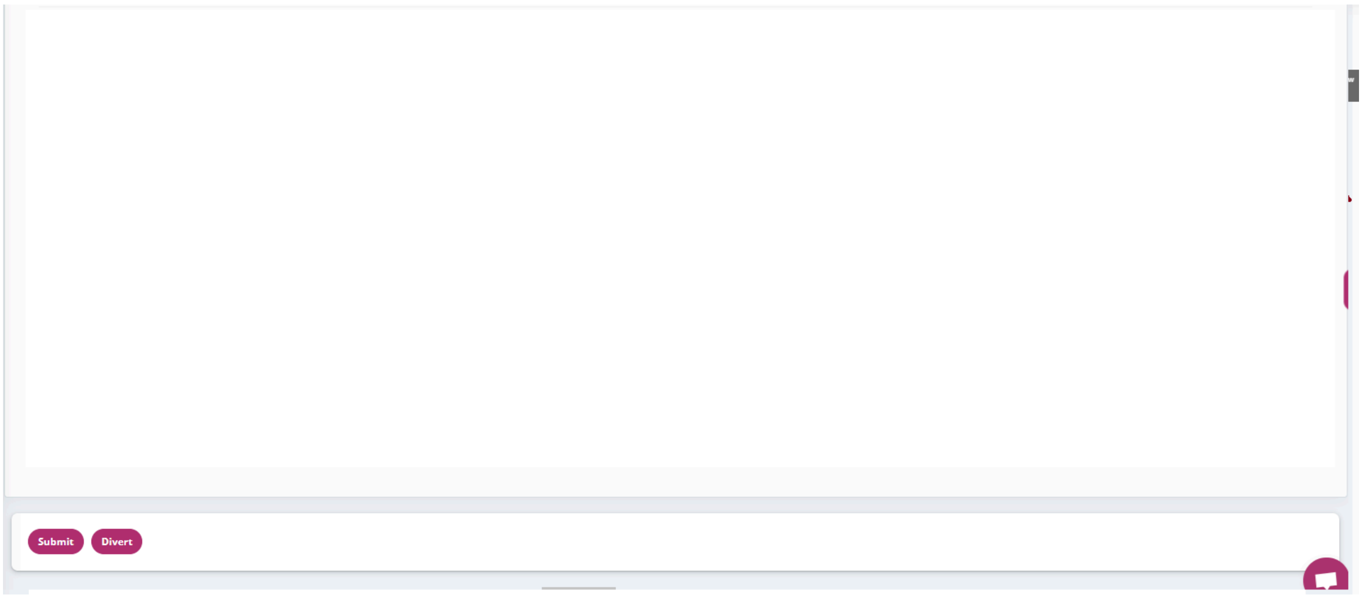
Process	Practice Patient Process	Initiated by	SYSTEM <SYSTEM>
Initiation date		Completion date	
Summary		Tags	

Form fields

Lets start with your email address

Notice: Upon duplication, the system will automatically redirect to the launch task of the "Create Patient" process.

Step 6: Load the Duplicated Instance



Notice: If the newly added decision and exclusive gateway are not positioned at the "Create Patient" launch task, the duplicated instances must be manually advanced through the process until reaching the task where Step 1 was implemented.

Step 7: Repeat Steps 3–6 as Required

Step 8: Revert the Modifications

(Remove the decision and exclusive gateway introduced in Step 1.)

Step 9: Save and Deploy the Reverted Process

Modifying Field Visibility in Tasks

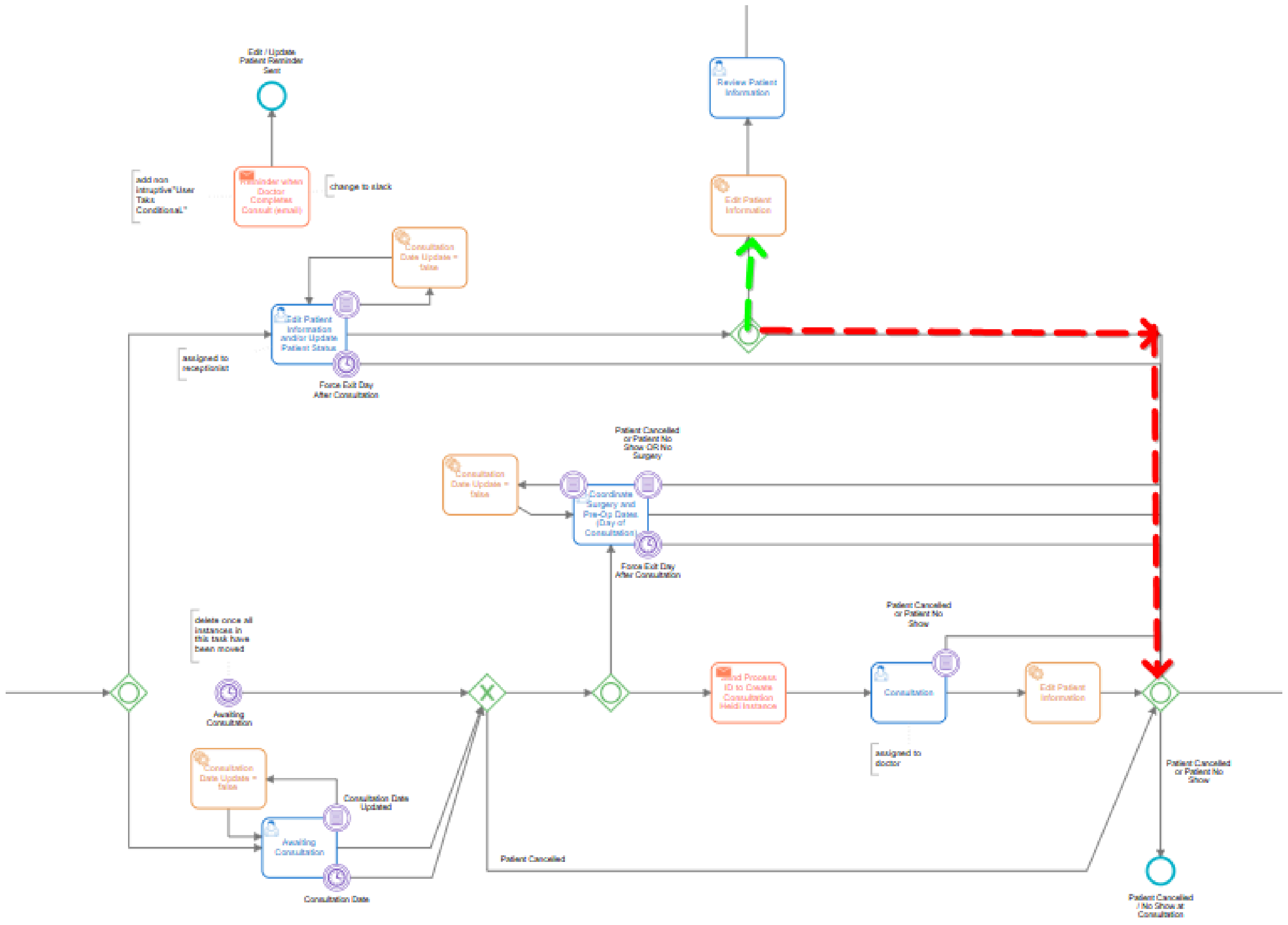
Field visibility can be adjusted within tasks. However, exercise caution regarding the following warnings:

- Warning: Field Visibility Preventing Script Execution
- Warning: Field Visibility Preventing User Interaction

Adding or Removing Scripts

Scripts may be added or removed as needed to enhance automation.

Warning: Parallel Tasks


Edit / Update Patient Reminder Sent
Review Patient Information
add non intruptive"User Task Conditional."
Reminder when Doctor Completes Consult (email)
change to slack
Edit Patient Information
Consultation Date Update = false
Edit Patient Information and/or Update Patient Status
assigned to receptionist
Force Exit Day After Consultation
Patient Cancelled or Patient No Show OR No Surgery
Consultation Date Update = false
Coordinate Surgery and Pre-Op Dates (Day of Consultation)
Force Exit Day After Consultation
delete once all instances in this task have been moved
Patient Cancelled or Patient No Show
Awaiting Consultation
Send Process ID to Create Consultation Heidi Instance
Consultation
Edit Patient Information
assigned to doctor
Consultation Date Update = false
Consultation Date Updated
Patient Cancelled or Patient No Show
Awaiting Consultation
Patient Cancelled
Consultation Date
Patient Cancelled / No Show at Consultation

Conditions cannot be applied to the red path, as it requires the loop to close unconditionally.

Initial Rollout: Task Assignment

During the first rollout, assign all tasks to yourself (as the administrator) and the intended user. This ensures you retain action rights beyond mere viewing privileges.

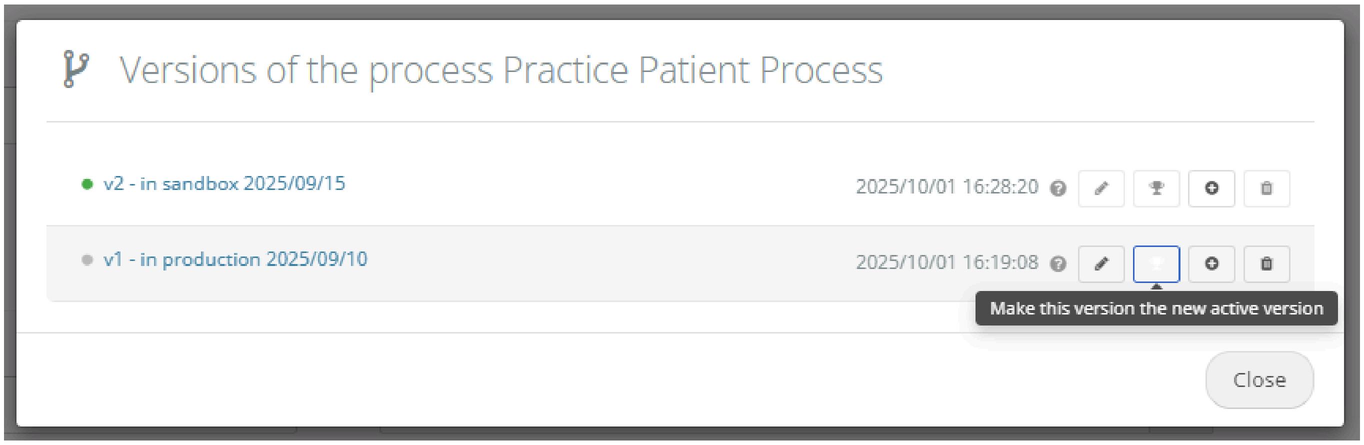
Example Scenario:
An LLM generates summaries from form fields and populates a text field in the backend. If results are visible in subsequent tasks, manual corrections may be required. Assigning tasks to yourself allows editing until the LLM prompt is refined.

Deploying a New Version

Staggered Deployment with Multiple Versions

Create internal tasks, such as "INTERNAL End of Deployment v{} Phase {}", at desired stages in version 1, assigned to the development team. When duplicating the flow for the next version, update blockers (e.g., rename to v{+1}). This facilitates management of stages and versions.

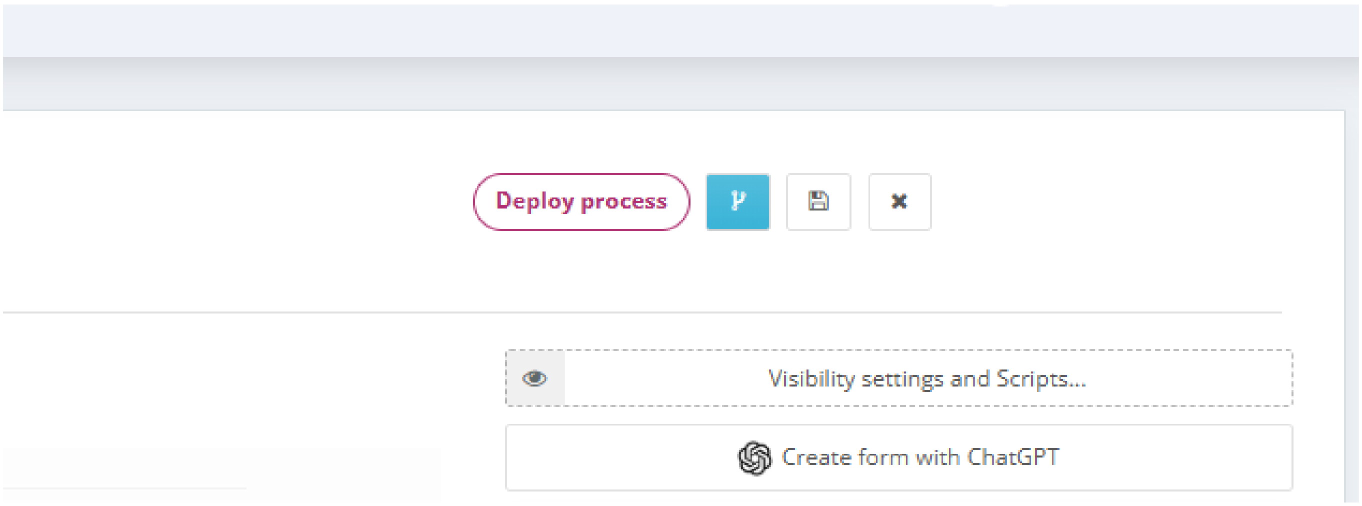
Step 1:


Versions of the process Practice Patient Process

● v2 - in sandbox 2025/09/15 2025/10/01 16:28:20

● v1 - in production 2025/09/10 2025/10/01 16:19:08

Step 2:



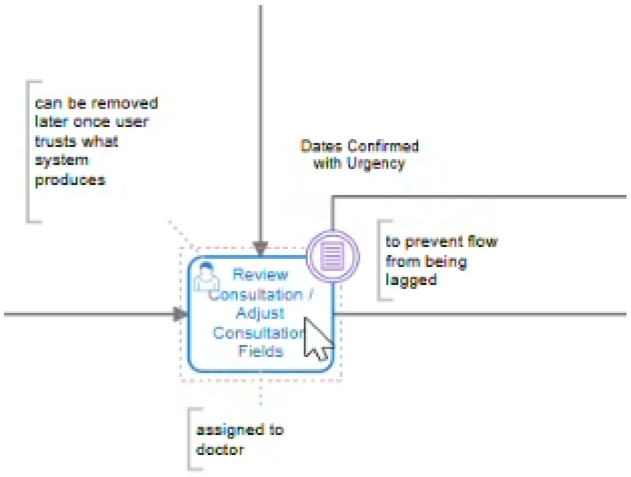
Save and deploy again after updates.

Risks Associated with Flow Changes

Create separate versions for changes and disable automatic migration of instances. Existing instances should remain on the old version, while new instances use the deployed version.

Prohibited Changes to Tasks

Changes such as removing conditional events or altering due dates cannot be made.


can be removed later once user trusts what system produces
Dates Confirmed with Urgency
to prevent flow from being lagged
Review Consultation / Adjust Consultation Fields
assigned to doctor

All instances in the affected task will fail to migrate. Modifications are only permissible if no instances exist in the task.

Additionally, transitioning from hidden to unhidden is inadvisable. Instead, assign the task to yourself in the current version and reassign to the final user in the new version.

Permitted Changes

Adding or removing decisions (including exclusive gateways) is allowed.

This is useful for reverting instances to a previous task or duplicating an instance to skip steps in the flow.

Warning: Changing Task Names with Script or Visibility Modifications

Avoid This Practice

Unlike form fields, where names can change while retaining identity (Information: System Handling of Field Name Changes), a different task name is treated as a new task.

If instances exist in a task and scripts or visibility are modified, these changes apply to those tasks.

However, if the task name is changed, modifications to scripts or field visibility do not propagate (instances retain the behavior of the previous name).

This is particularly problematic for new scripts or form fields, and critical if new fields are used in gateways. Introduce an additional task immediately after to set the new field correctly, as it will not appear in the renamed task.

When to Proceed

1. No concurrent changes to scripts or fields.
2. No existing instances in the task (or none created during updates).

Note: Untested Approach

Potential Workaround

If instances exist in a task requiring name and functional updates (scripts or visibility):

1. Apply script or visibility changes to the task as-is.
2. Use real-time reports to verify application to existing instances.
3. If confirmed, proceed with the name change.

This ensures existing instances receive functional updates (without name change), while new instances get the updated name and functions.

Warning: Using End-of-Process Elements

Reference: Common Errors

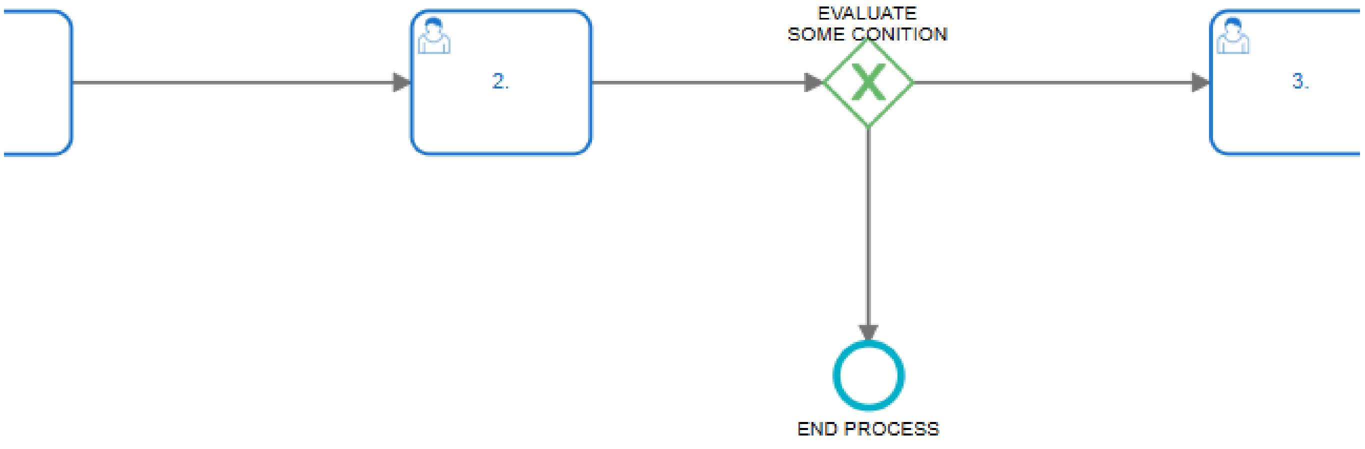
Key Consideration

The end-of-process only registers when the last token arrives, and real-time reports reflect the name of the last completed end-of-process.

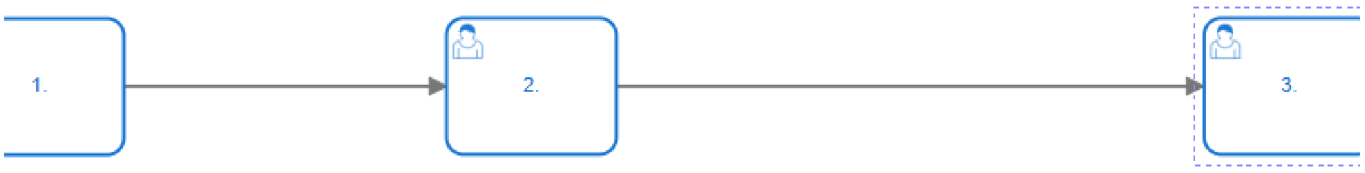
Notice: Conditions often rely on form fields. Deleting such fields overcomes issues but erases data from prior instances.

With Exclusive Gateways

A decision follows Task 2: If a condition is met, proceed to Task 3; otherwise, end the process.



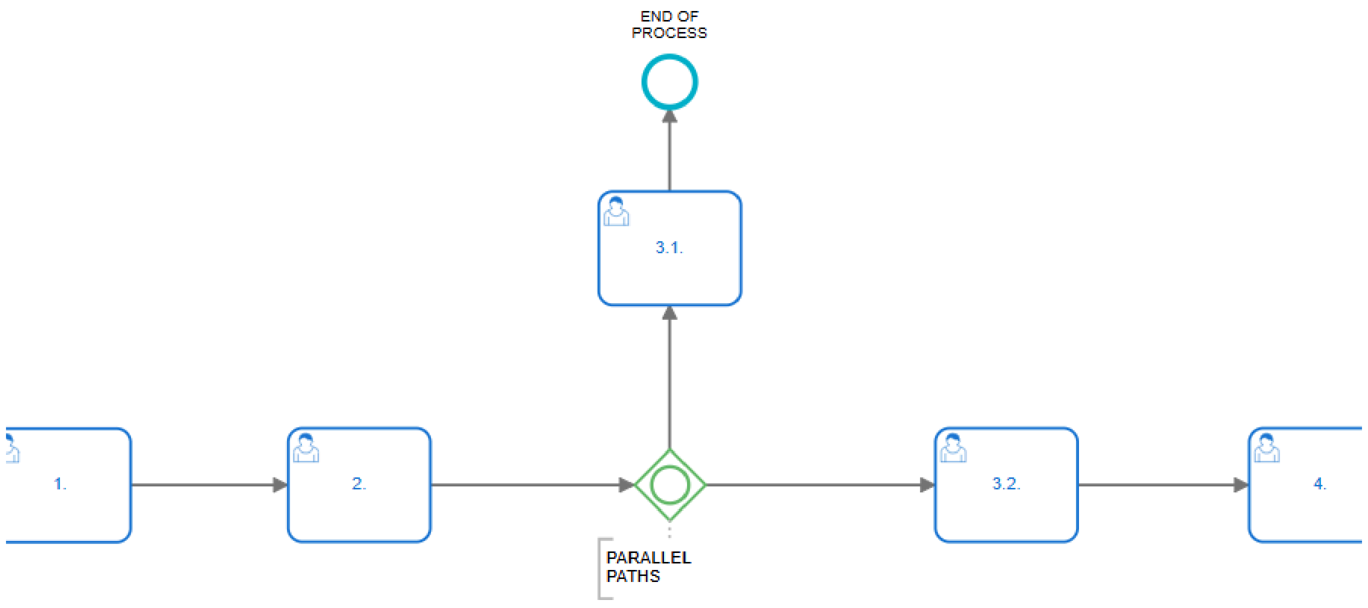
For instances (e.g., FLOW-1, FLOW-2) created before an update removing the gateway:



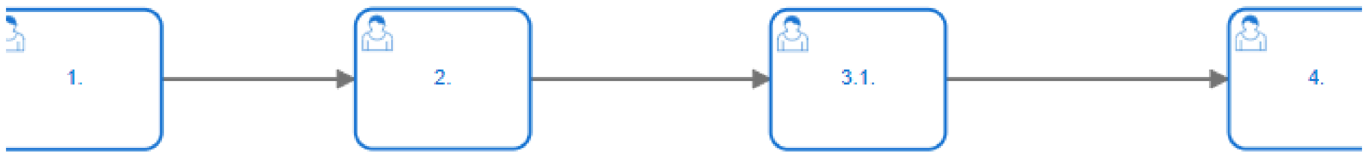
Post-Task 2 completion, if the prior condition is met, they end prematurely instead of proceeding to Task 3. New instances follow the updated flow (1 → 2 → 3).

With Inclusive Gateways

Instances post-Task 2 split to Tasks 3.1 (ending) and 3.2 (continuing to 4).

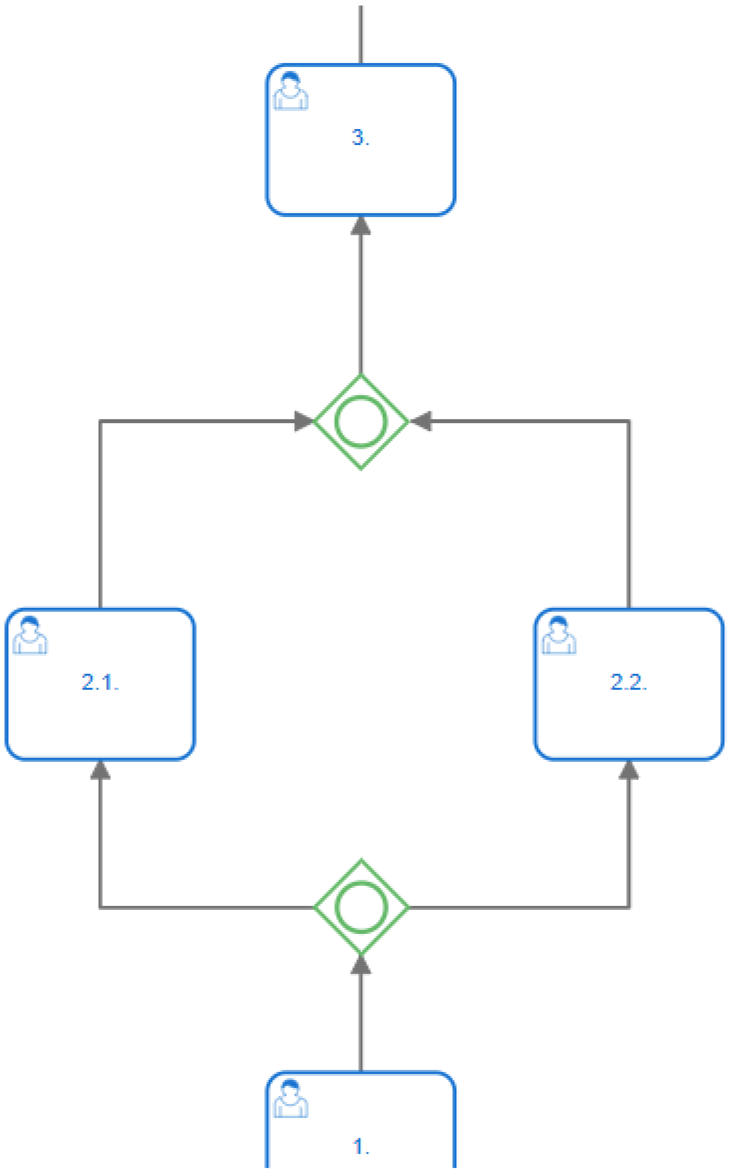


For pre-update instances (FLOW-1, FLOW-2), after update removing the gateway:

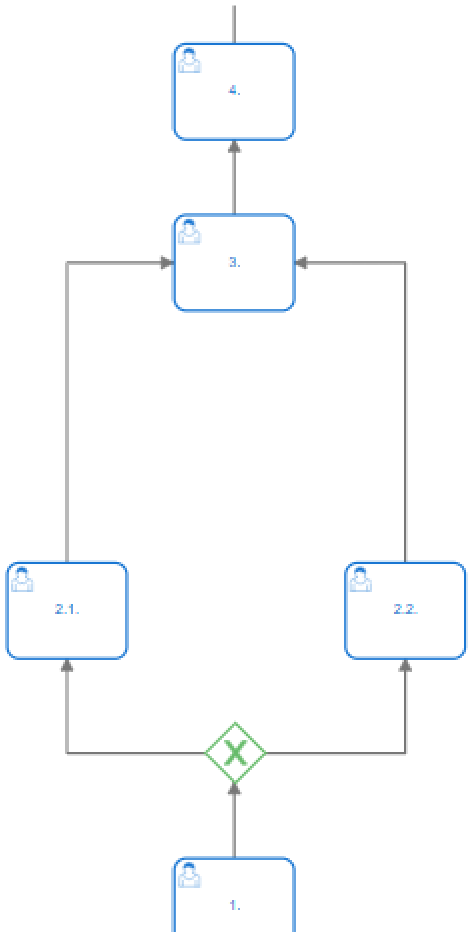


They end after Task 2 instead of continuing.

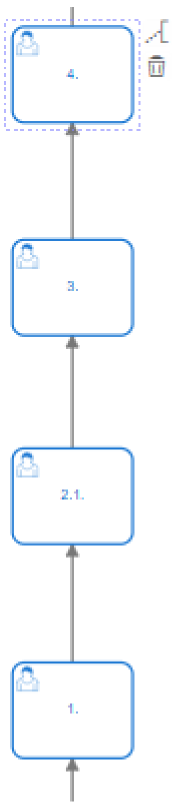
Anomaly Explanation: Changes without end-of-process work, even with instances in deleted tasks (e.g., 2.2). However, end-of-process involvement ignores flow updates.



or



to



Warning: Field Visibility Preventing Script Execution

Set all fields involved in scripts to "Editable" by default at the task, then use scripts to adjust to hidden or read-only.

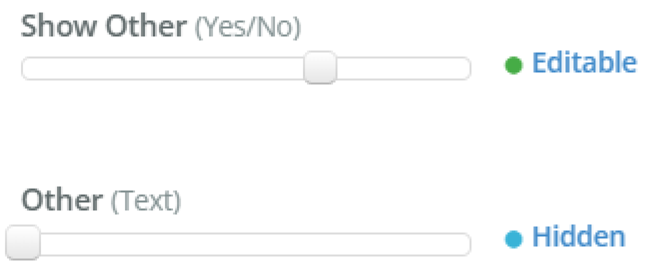
Rationale:

1. Scripts can retrieve values from read-only fields (`.getFieldValue`) but cannot update them (`.setFieldValue`).
2. Scripts cannot retrieve or update hidden fields.

Warning: Field Visibility Preventing User Interaction

Hidden Field Required by Script

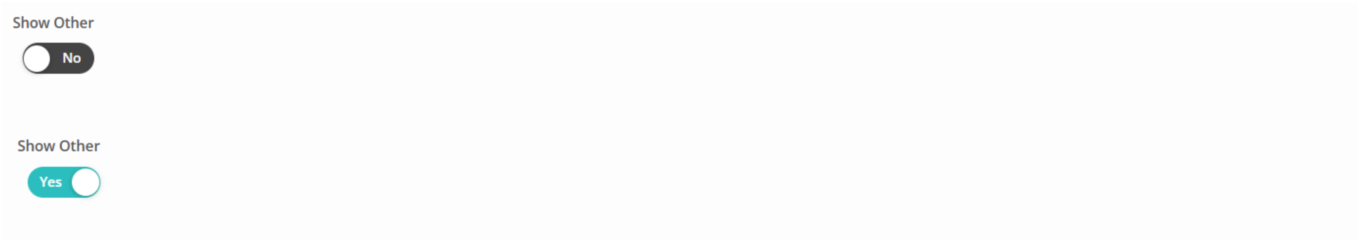
Task Configuration: Field set to hidden.



Expected Script Behavior: Make field required (fails).

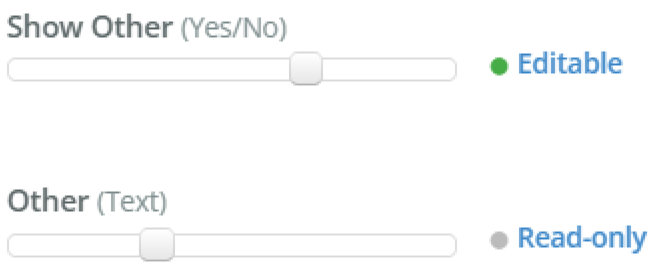
```
function showField(){
    var show = Flokzu.getFieldValue([[Show Other]]);
    if (show === true){Flokzu.setRequired([[Other]])}
    // else {Flokzu.setHidden([[Other]])}
}
Flokzu.onInit(showField);
Flokzu.onChange([[Show Other]], showField);
```

Post-Deployment:



Read-Only Field Required by Script

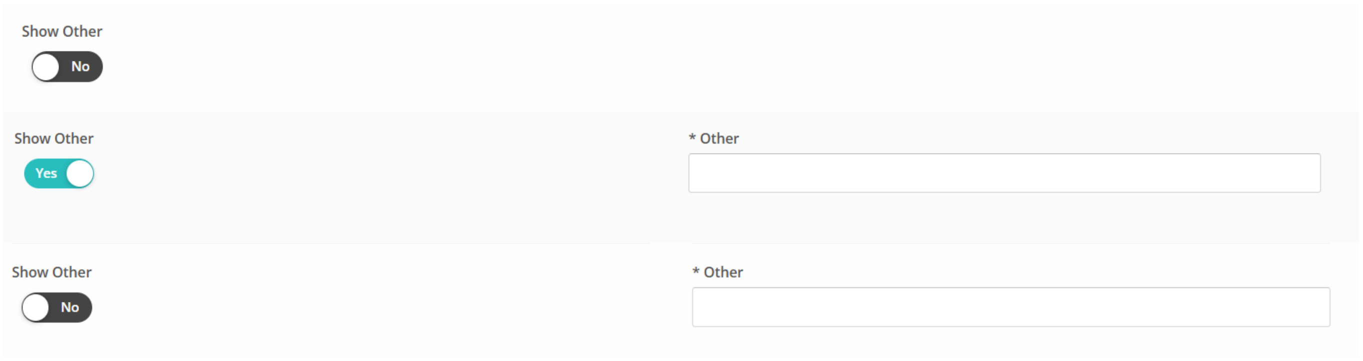
Task Configuration: Field set to read-only.



Expected Script Behavior: Make field required (partially succeeds).

```
function showField(){
    var show = Flokzu.getFieldValue([[Show Other]]);
    if (show === true){Flokzu.setRequired([[Other]])}}
Flokzu.onInit(showField);
Flokzu.onChange([[Show Other]], showField);
```

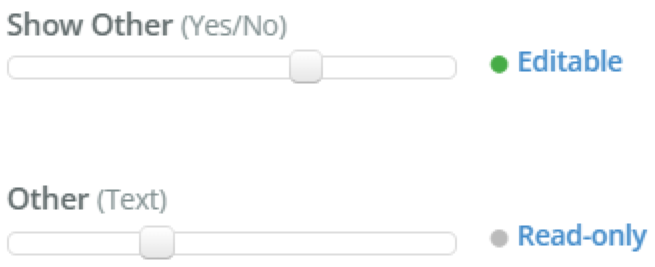
Post-Deployment:



(May result from lacking an else clause to reset visibility, as demonstrated below.)

Read-Only Field Hidden or Required by Script

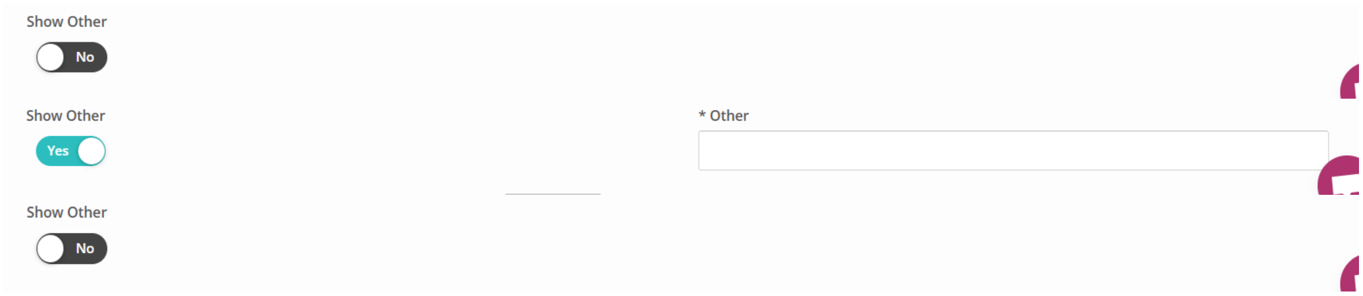
Task Configuration: Field set to read-only.



Expected Script Behavior: Make required or hide (succeeds).

```
function showField(){
    var show = Flokzu.getFieldValue([[Show Other]]);
    if (show === true){Flokzu.setRequired([[Other]])}else {
Flokzu.setHidden([[Other]]) }
}
Flokzu.onInit(showField);
Flokzu.onChange([[Show Other]], showField);
```

Post-Deployment:



(Functions with every change; requires else clause.)

Notice: Similar behavior occurs with "Editable" settings.

Best Practice: Set all script-involved fields to editable; let scripts control hidden, read-only, or required states. Use read-only if only fetching values (`.getFieldValue`). If planning read-only via script, set it directly in the task to save effort; for hidden, leave editable and hide via script.

Configuring External Forms for Interaction

External forms can auto-populate when initialized from the main process.

In the Main Form

1. Set all involved fields to editable.
   Note: For fields using only `.getFieldValue` and planned as hidden, set hidden at the step to avoid duplication.
2. Use a script to update a field with the external form link.

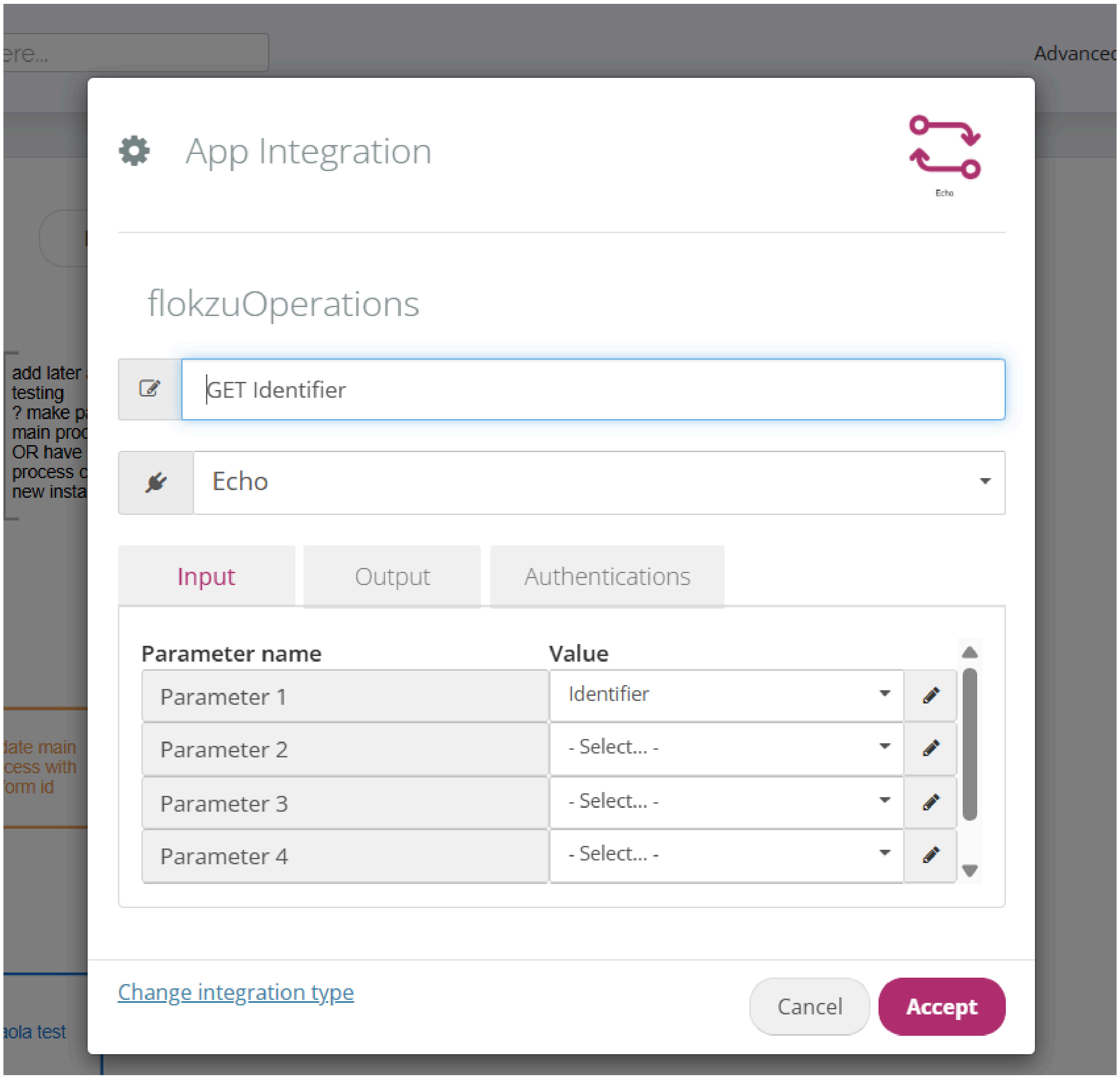
```
function externalForm(){
    var id = $('#fkz_ref').text();
    var number = Flokzu.getFieldValue([[Number]]);
    var name = Flokzu.getFieldValue([[Name]]);
    var surname = Flokzu.getFieldValue([[Surname]]);
    var fullname = name + ' ' + surname;
    var first = 'https://app.flokzu.com/public/----?Number=';
    var second = '&Name=';
    var third = '&ID=';
    var URL = first.concat(number, second, fullname, third, id);
    // ALTERNATIVE FORMAT using text field instead of weblink
    // required for cases where https: contains blank spaces
    // var link = '<a href="' + URL + '">Name of Link</a>';
    Flokzu.setFieldValue([[External Link]], URL); //
Flokzu.setFieldValue([[External Link]], link);
    // Flokzu.setHidden([[External Link]]); // POST-TESTING
}

Flokzu.onInit(externalForm);
// Flokzu.onChange([[ ]], externalForm);
```

In the External Form

3. Create required fields for URL population (set to editable).
4. Include an integer field for the external form's process identifier (hide it; backend can update hidden fields).
5. Retrieve fields from URL via script.

```
// EXAMPLE
// https://app.flokzu.com/public/----?Number=123&Name=Me&ID=3

$.urlParam = function(name){ // dont touch!!!!
    var results = new RegExp('[\?&]' + name + '=
([^&#]*)').exec(window.location.href);
    console.log(results);
    if (results == null){
       return null;
    }
    else {
       return decodeURI(results[1]) || 0;
    }
}

function Defaultfields(){
    var number=  $.urlParam('Number');
    var name=  $.urlParam('Name');
    var id = $.urlParam('ID');

    console.log(number);

    Flokzu.setFieldValue([[Number]], number);
    // Flokzu.setReadOnly([[Number]]); // POST-TESTING

    Flokzu.setFieldValue([[Name & Surname]], name);
    // Flokzu.setReadOnly([[Name & Surname]]); // POST-TESTING

    Flokzu.setFieldValue([[Main Process ID]], id);
    // Flokzu.setHidden([[Main Process ID]]); // POST-TESTING
}

Flokzu.onInit(Defaultfields);
```

6. Obtain the form ID.



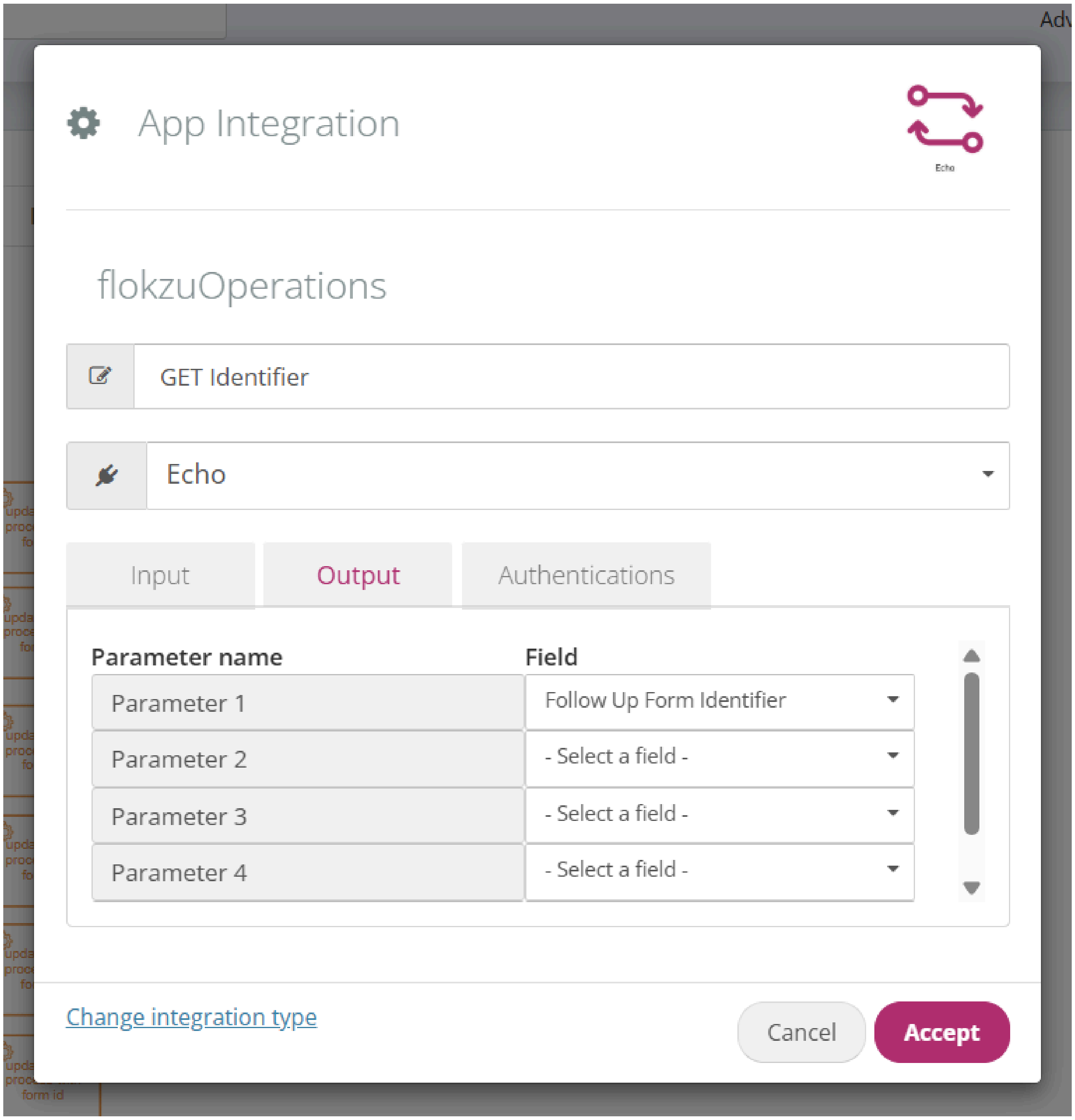
App Integration

flokzuOperations

GET Identifier

Echo

Parameter name	Value
Parameter 1	Identifier
Parameter 2	- Select... -
Parameter 3	- Select... -
Parameter 4	- Select... -


App Integration

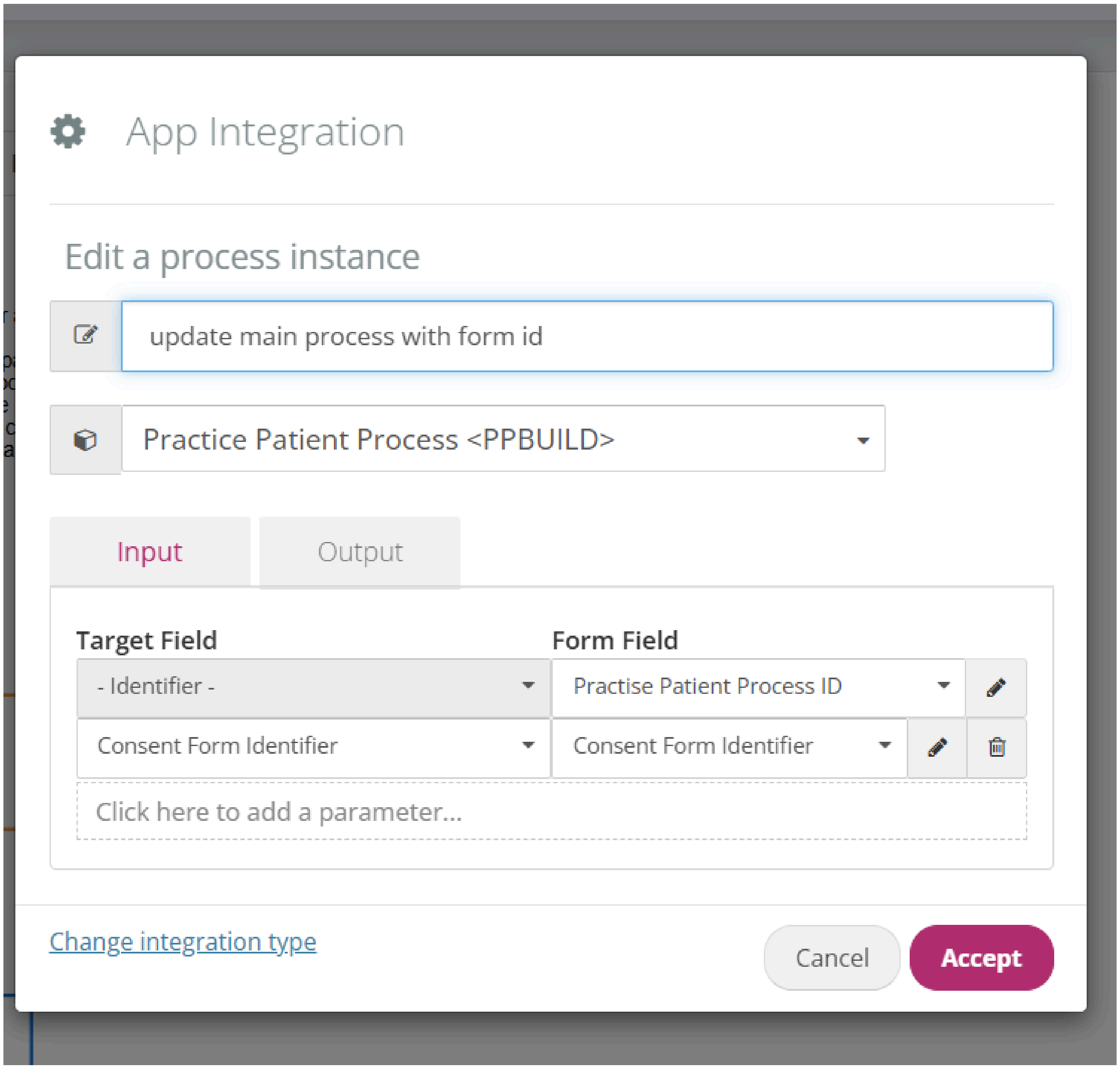
flokzuOperations

GET Identifier

Echo

Parameter name	Field
Parameter 1	Follow Up Form Identifier
Parameter 2	- Select a field -
Parameter 3	- Select a field -
Parameter 4	- Select a field -

7. Update the main process.



App Integration

Edit a process instance

update main process with form id

Practice Patient Process <PPBUILD>

Target Field	Form Field
- Identifier -	Practise Patient Process ID
Consent Form Identifier	Consent Form Identifier

Testing

8. Create an instance and advance to the relevant task.
9. Verify the main form's external link field updates.
10. Open the link and confirm auto-population.
11. Complete the external form.
12. Refresh the main task and verify the external instance ID updates.
13. Post-testing, update scripts in In the Main Form and In the External Form to set hidden/read-only as needed.

Field Visibility Impact on Script Execution and User Interaction

Fields Updated by Scripts

Must be editable at the task where the script runs.

Conditionally Visible Fields

Must be editable at the task where the script runs. Hidden fields cannot be made required by scripts (prevents user interaction). For fields appearing based on other values, set editable and hide via script.

Fields for Value Retrieval by Scripts

Must be editable or read-only, then hidden on initialization.

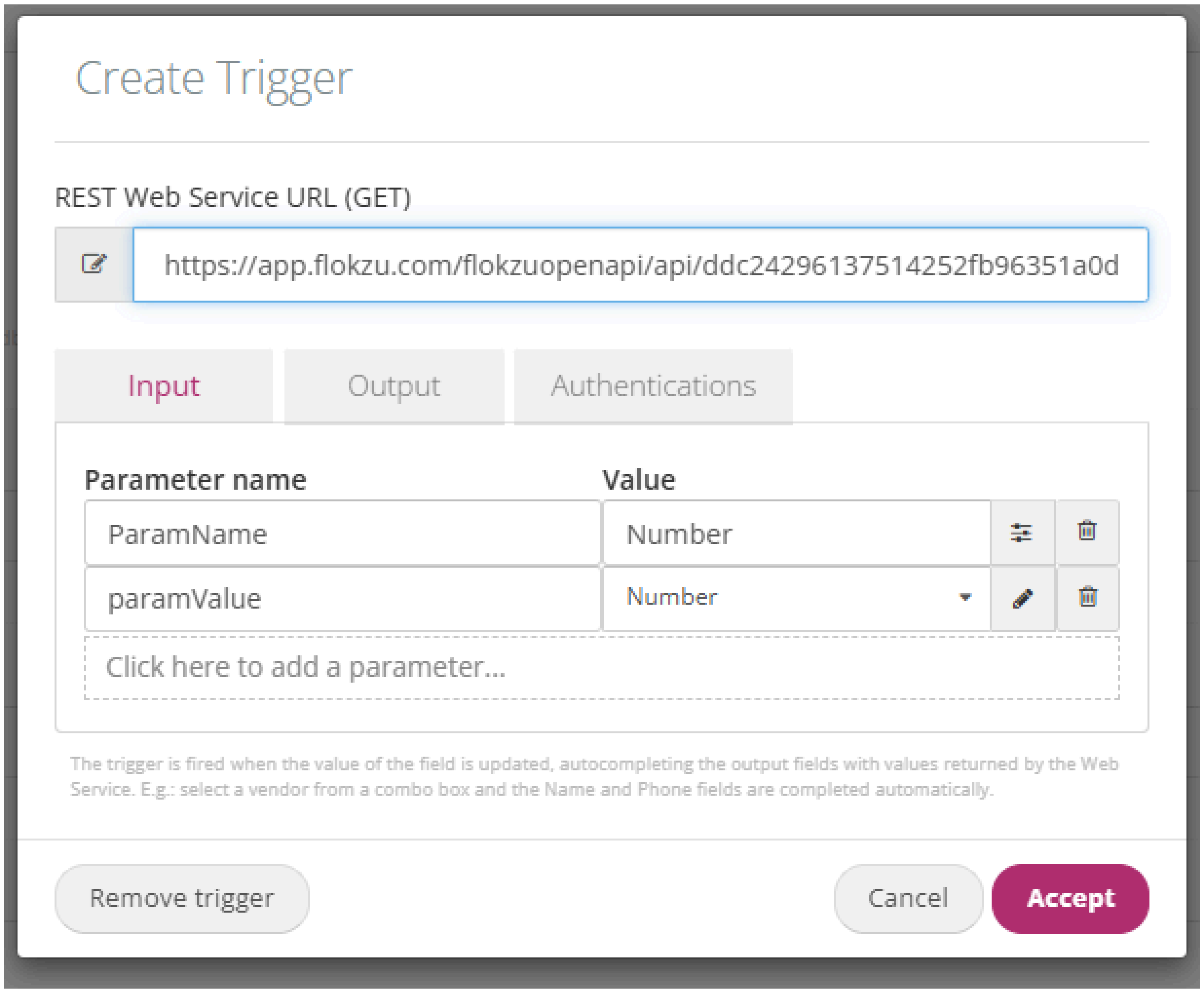
Fields Updated from Databases

Can remain hidden.

Configuring Database Interactions from Form Fields (Triggers)

Input

Lookup values from the database (use unique identifiers).



Create Trigger

REST Web Service URL (GET)

https://app.flokzu.com/flokzuopenapi/api/ddc24296137514252fb96351a0d

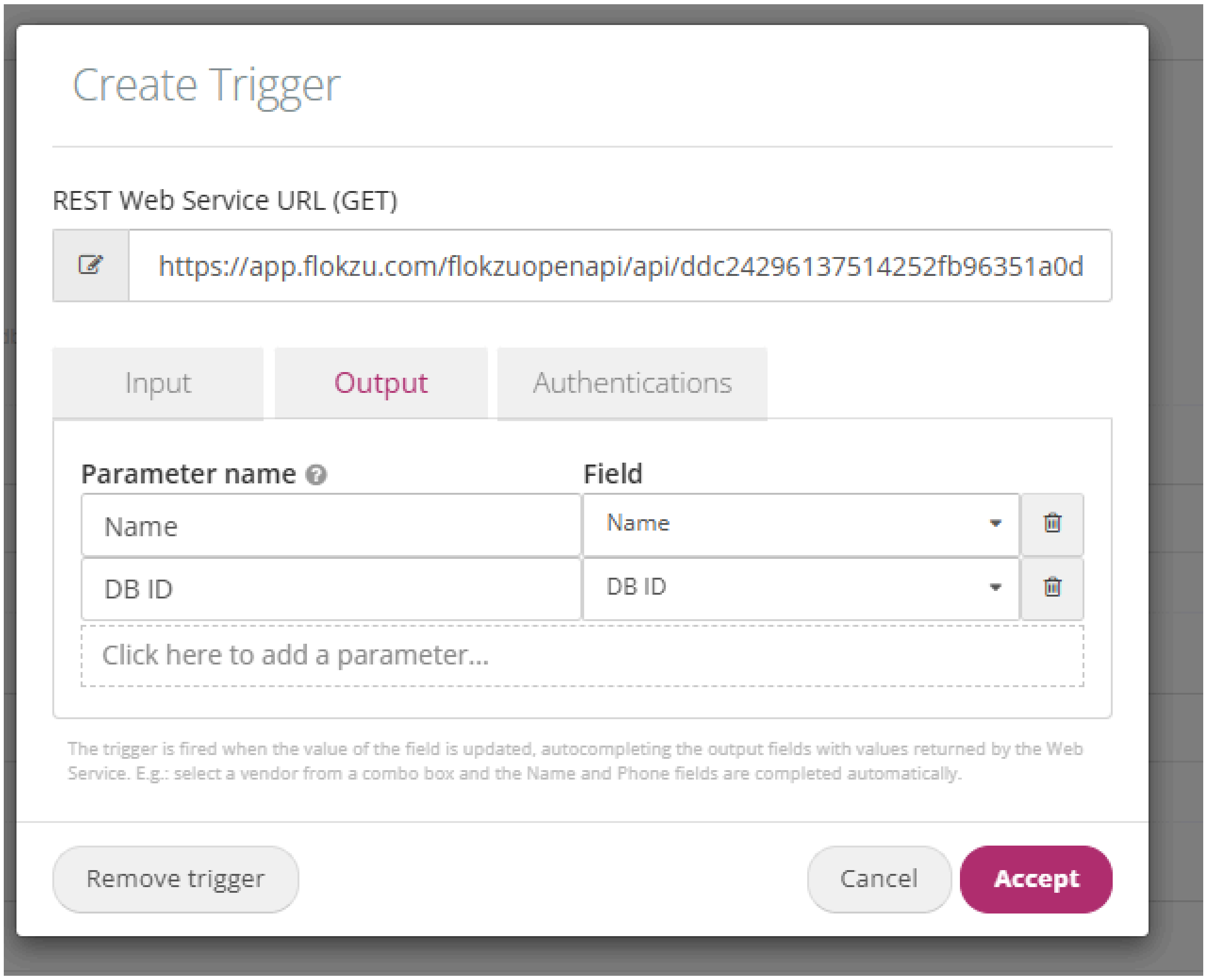
Parameter name	Value
ParamName	Number
paramValue	Number

The trigger is fired when the value of the field is updated, autocompleting the output fields with values returned by the Web Service. E.g.: select a vendor from a combo box and the Name and Phone fields are completed automatically.

(Configured in the "Number" field.)

Output

Populate fields accordingly.



Create Trigger

REST Web Service URL (GET)

https://app.flokzu.com/flokzuopenapi/api/ddc24296137514252fb96351a0d

Parameter name	Field
Name	Name
DB ID	DB ID

The trigger is fired when the value of the field is updated, autocompleting the output fields with values returned by the Web Service. E.g.: select a vendor from a combo box and the Name and Phone fields are completed automatically.

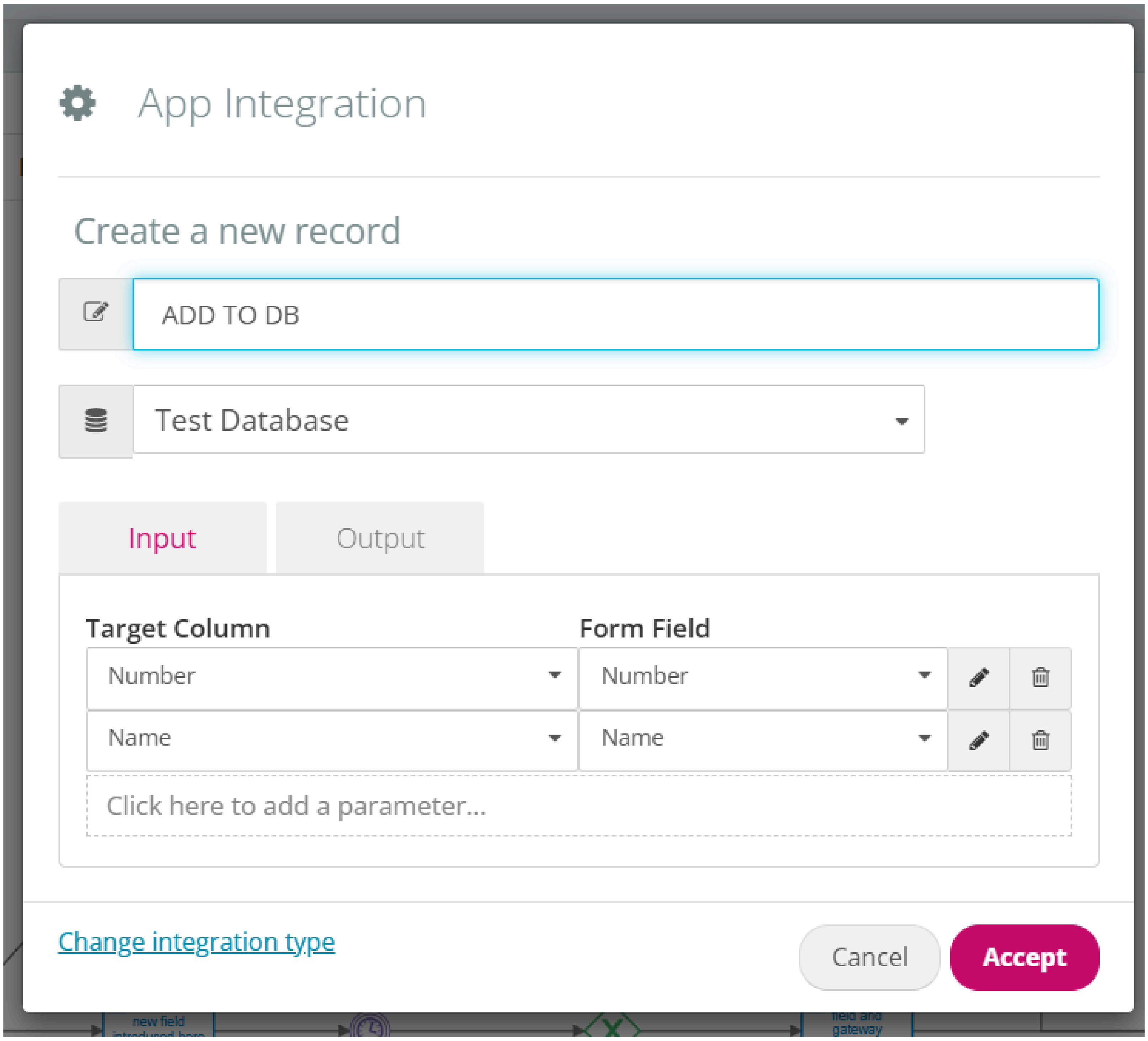
Note: Parameter names must match database column names exactly.

Warning: Populating the database ID field this way fails. Use And Return Database ID to Form instead. This flow (Add to / Edit Record in Database) works but logs errors for new entries (no matching record). It functions even without a unique identifier, adding records without checks.

Adding and Editing Database Records Post-Task

Notice: Fields storing database primary keys must be integers and can remain hidden.

Add Record to Database



App Integration

Create a new record

ADD TO DB

Test Database

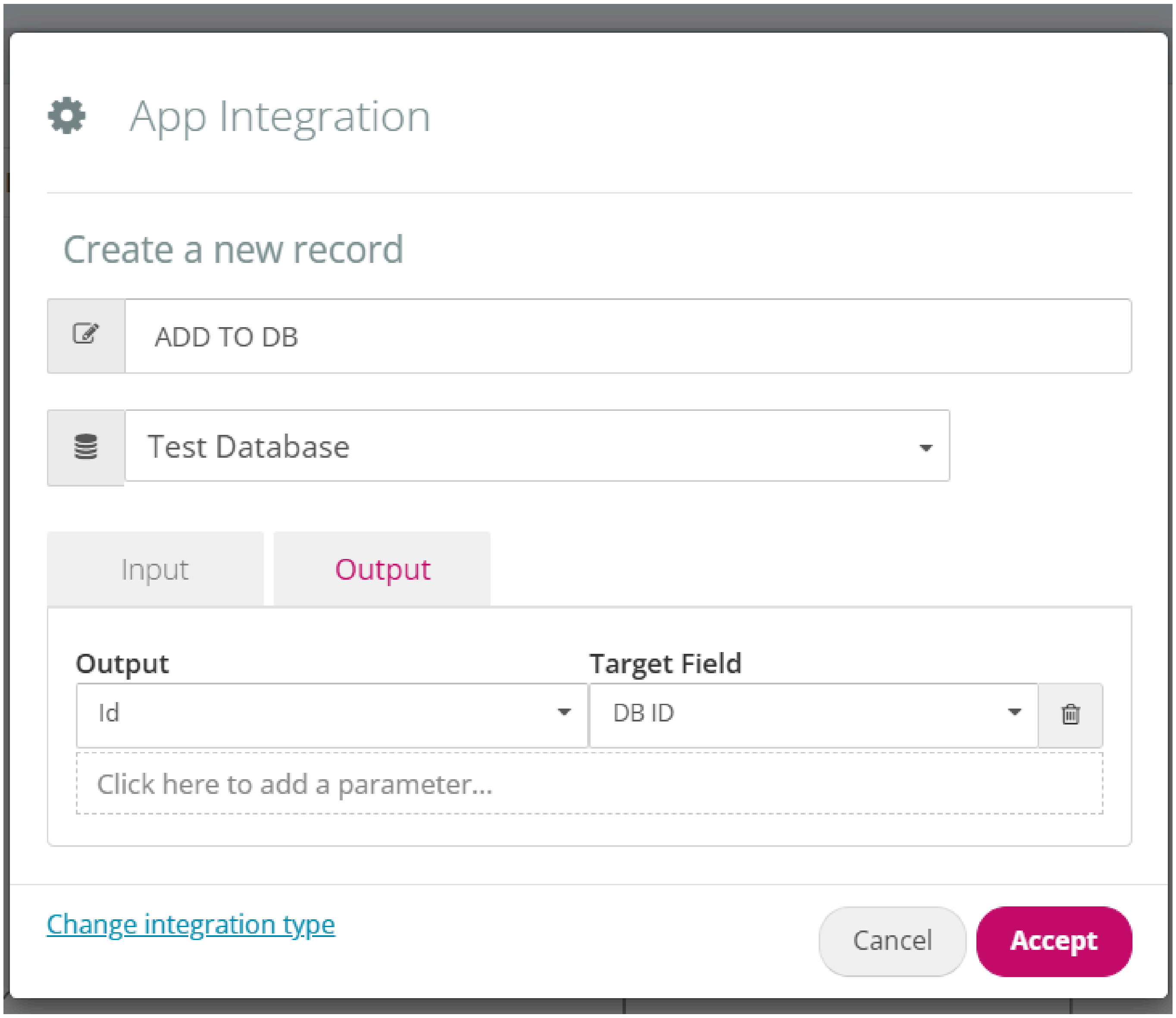
Input | Output

Target Column	Form Field
Number	Number
Name	Name

Click here to add a parameter...

And Return Database ID to Form

Field can be hidden.


App Integration

Create a new record

ADD TO DB

Test Database

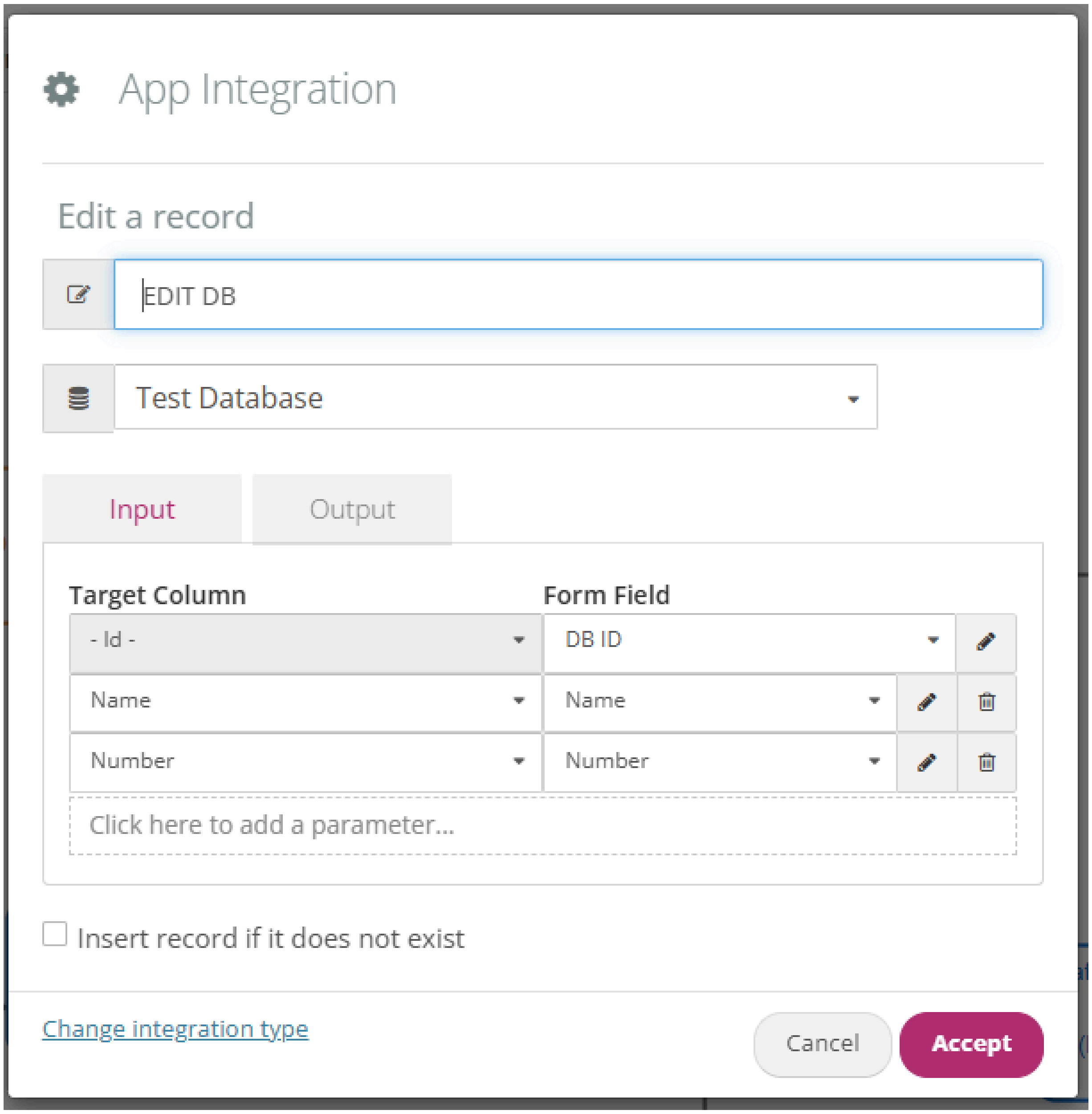
Input | Output

Output	Target Field
Id	DB ID

Click here to add a parameter...

Edit Record Based on Database ID

(Using And Return Database ID to Form.)


App Integration

Edit a record

EDIT DB

Test Database

Input | Output

Target Column	Form Field
- Id -	DB ID
Name	Name
Number	Number

Click here to add a parameter...

☐ Insert record if it does not exist

Change integration type

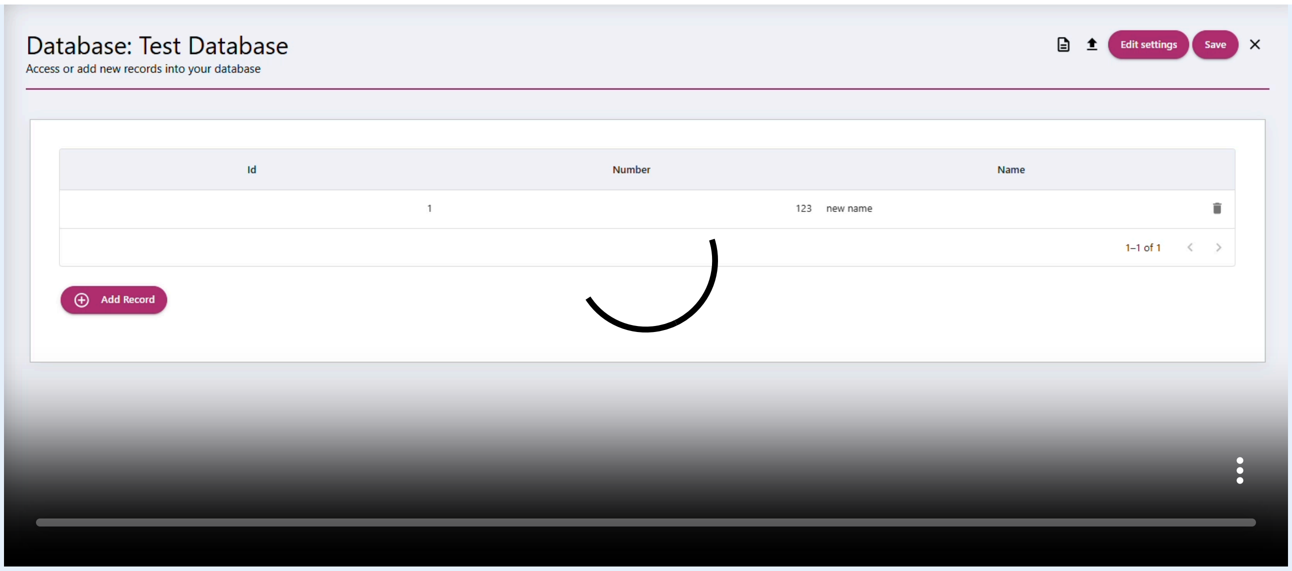
Cancel | Accept

Add to / Edit Record in Database

Since Output fails to update ID fields (even if editable), use this flow.

Mechanism

New Record: Unique identifier not found in Fetch Database ID (If Exists); ID field remains blank. Add / Edit Record (Depending on If Database Record ID Exists) adds the record. Populate Database ID Field in Form sets the new ID, usable for edits in Edit Record Based on Database ID.


Database: Test Database

Access or add new records into your database

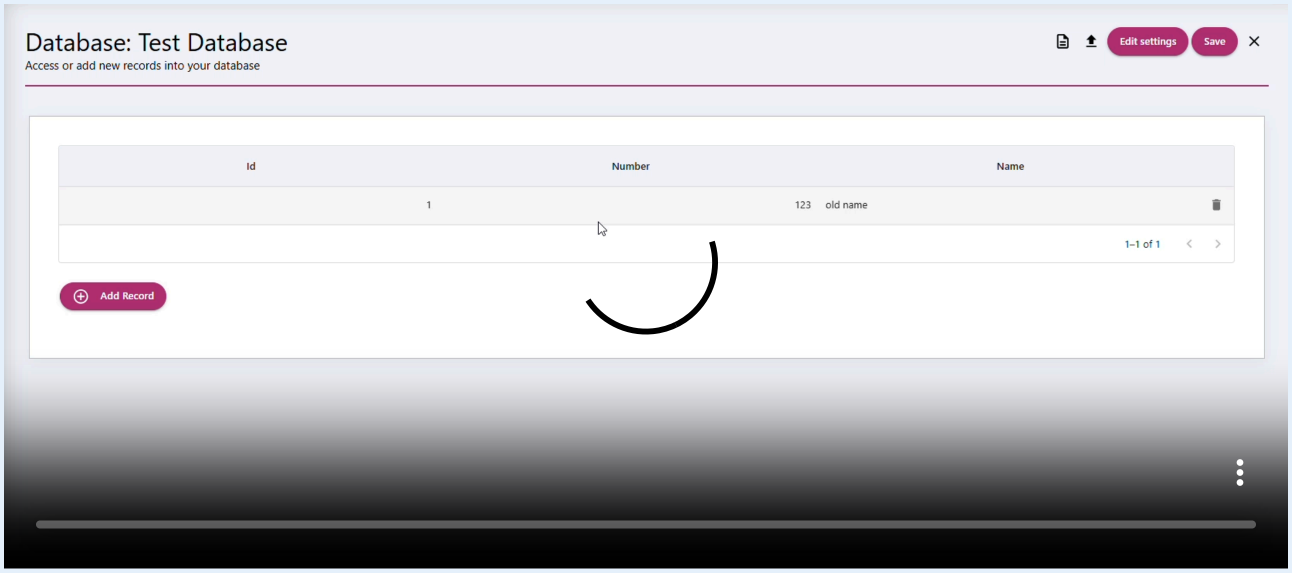
Edit settings | Save

Id	Number	Name
1	123	new name

1–1 of 1

Add Record

Existing Record: Unique identifier found; ID populated via [And Populate Database ID Form Field](). Edits occur in [Add to / Edit Record in Database]() (no-op if unchanged), allowing changes at the same point as new records via [Edit Record Based on Database ID]().


Database: Test Database

Access or add new records into your database

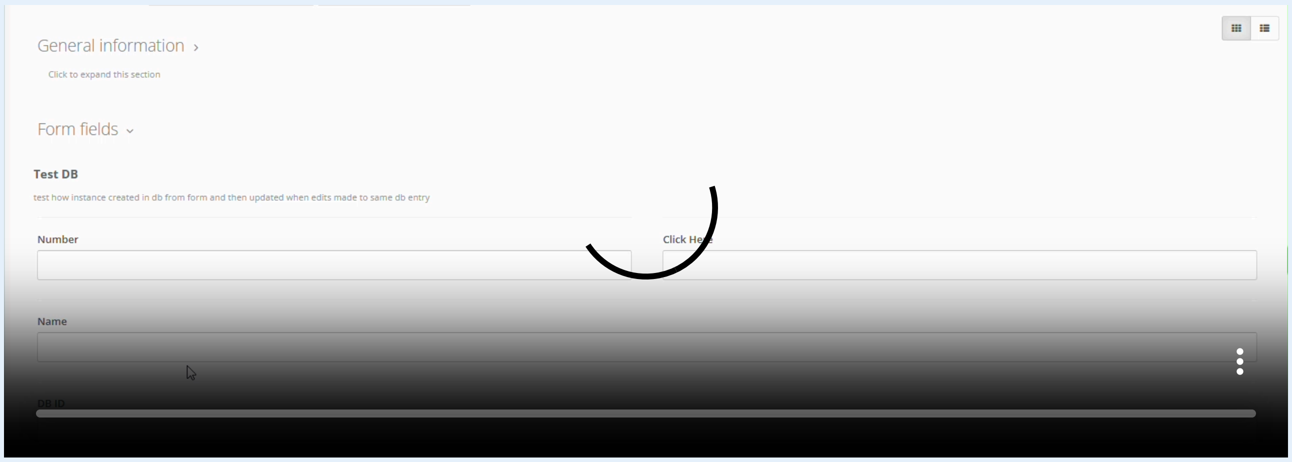
Edit settings | Save

Id	Number	Name
1	123	old name

1–1 of 1

Add Record

Notice: Updates ID even if unique identifier is null.


General information ›

Click to expand this section

Form fields ⌄

Test DB

test how instance created in db from form and then updated when edits made to same db entry

Number

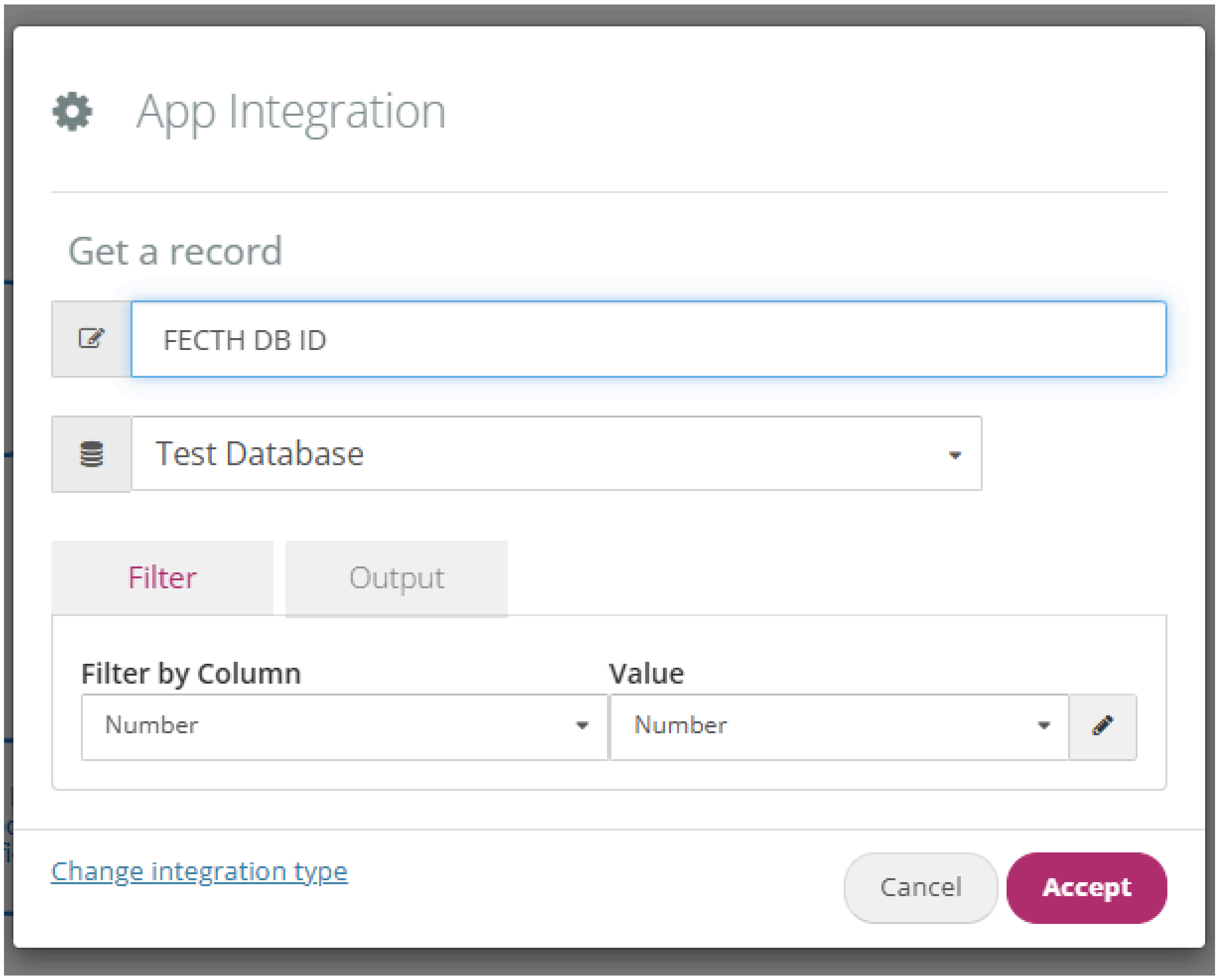
Click Here

Name

DB ID

Fetch Database ID (If Exists)

Use the same unique identifier field as in Input.



App Integration

Get a record

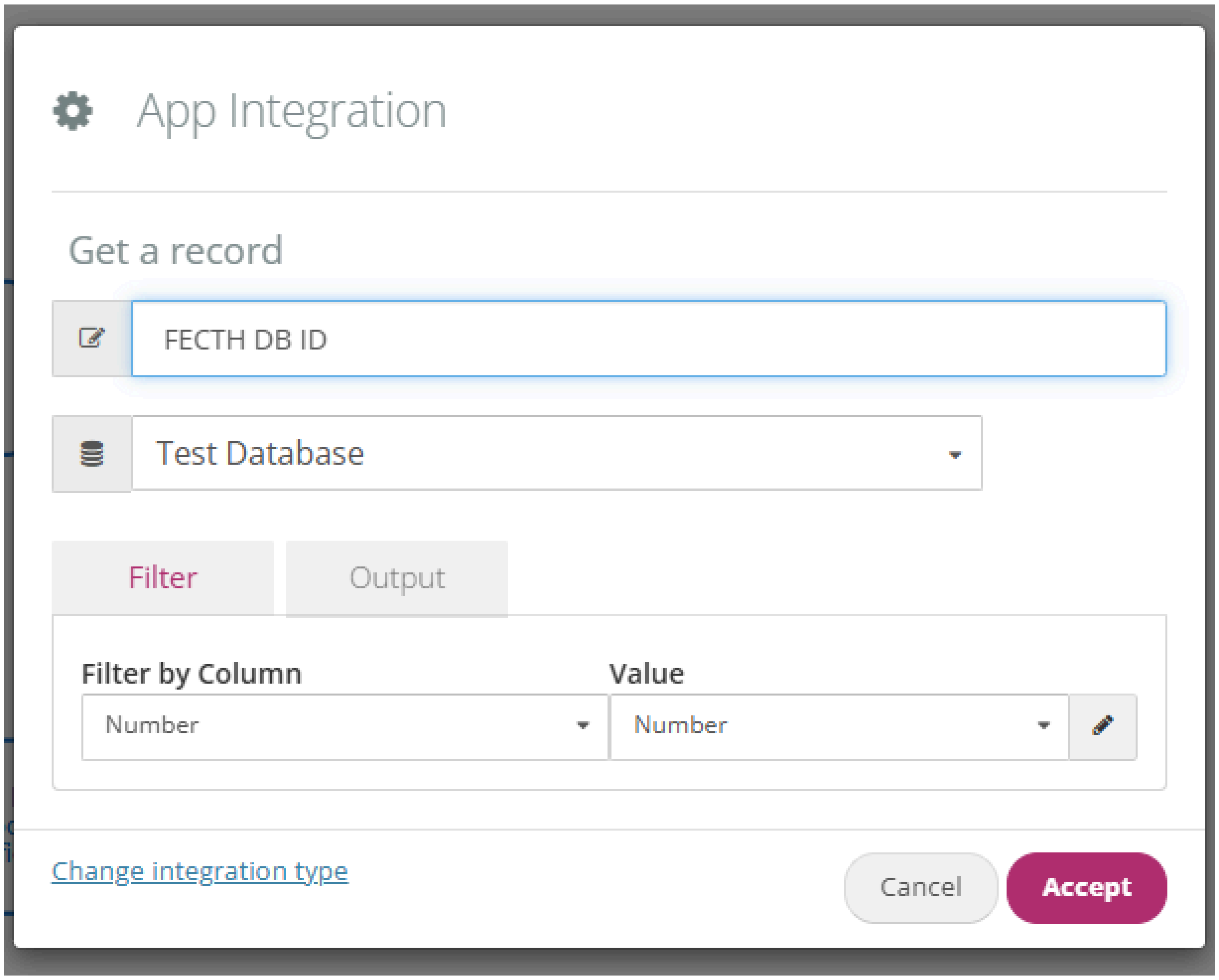
FECTH DB ID

Test Database

Filter by Column: Number

Value: Number

And Populate Database ID Form Field



App Integration

Get a record

FECTH DB ID

Test Database

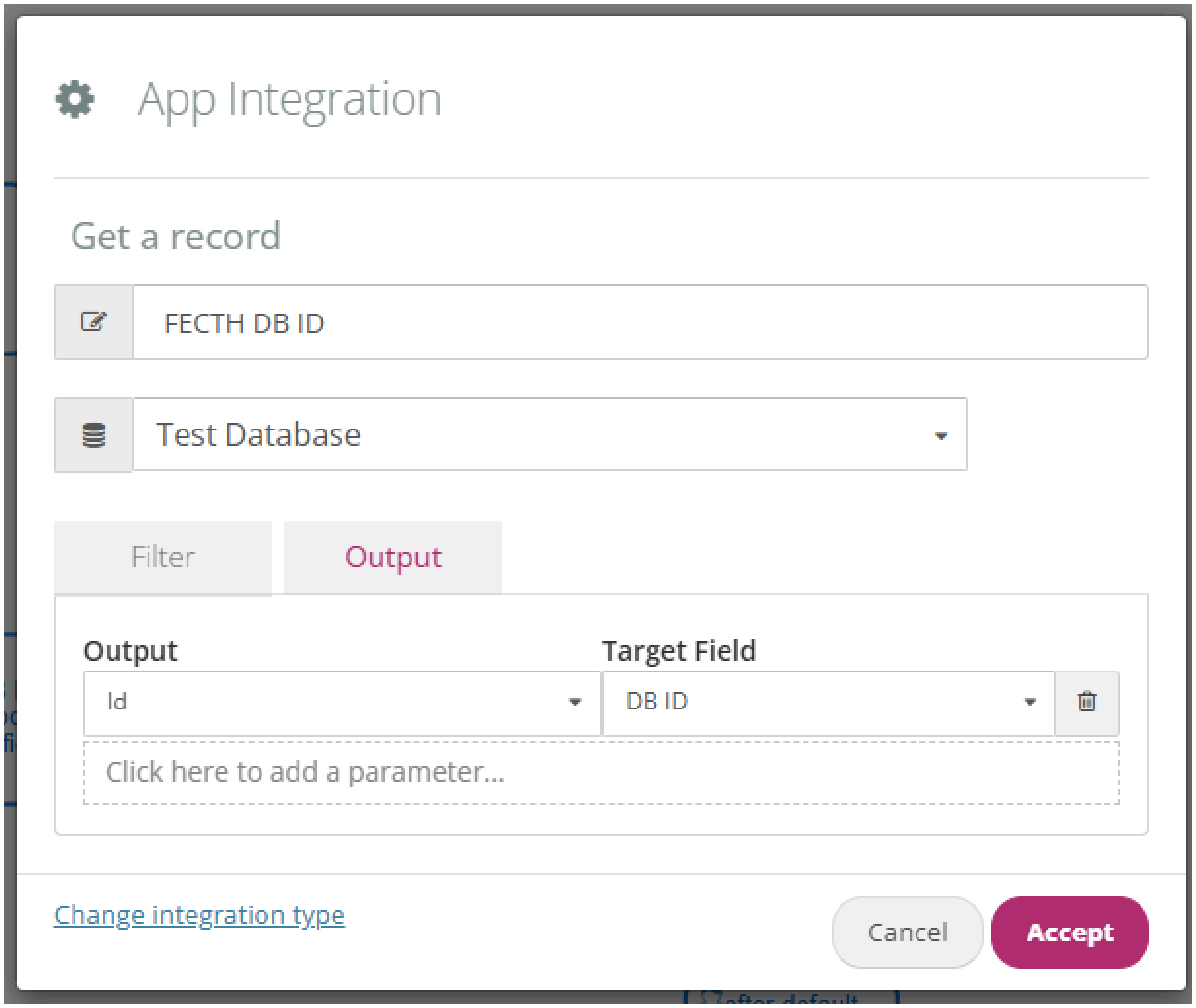
Filter | Output

Filter by Column: Number

Value: Number

Change integration type

Cancel | Accept


App Integration

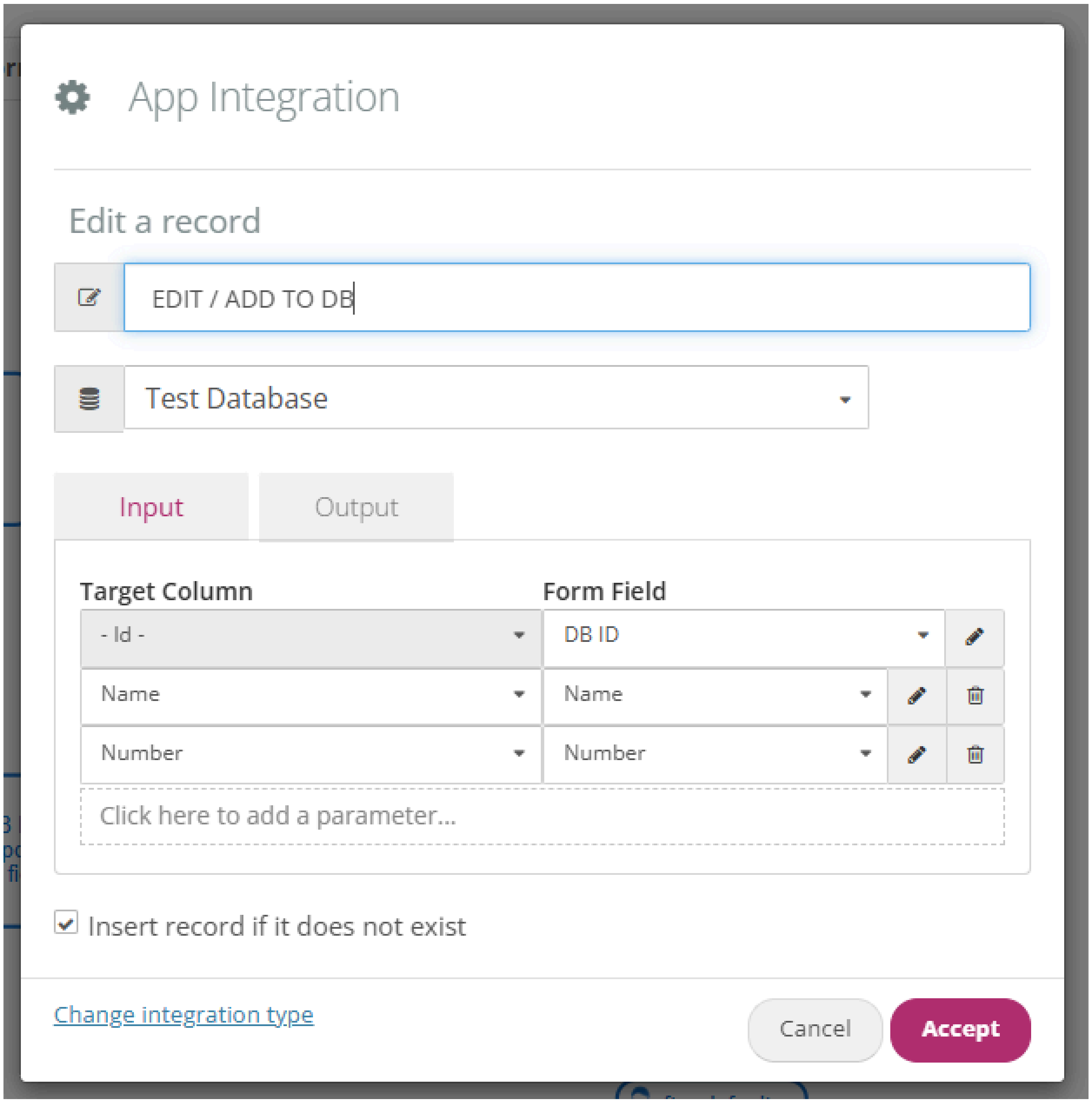
Get a record

FECTH DB ID

Test Database

Output	Target Field
Id	DB ID

Add / Edit Record (Depending on If Database Record ID Exists)


App Integration

Edit a record

EDIT / ADD TO DB

Test Database

Input | Output

Target Column	Form Field
- Id -	DB ID
Name	Name
Number	Number

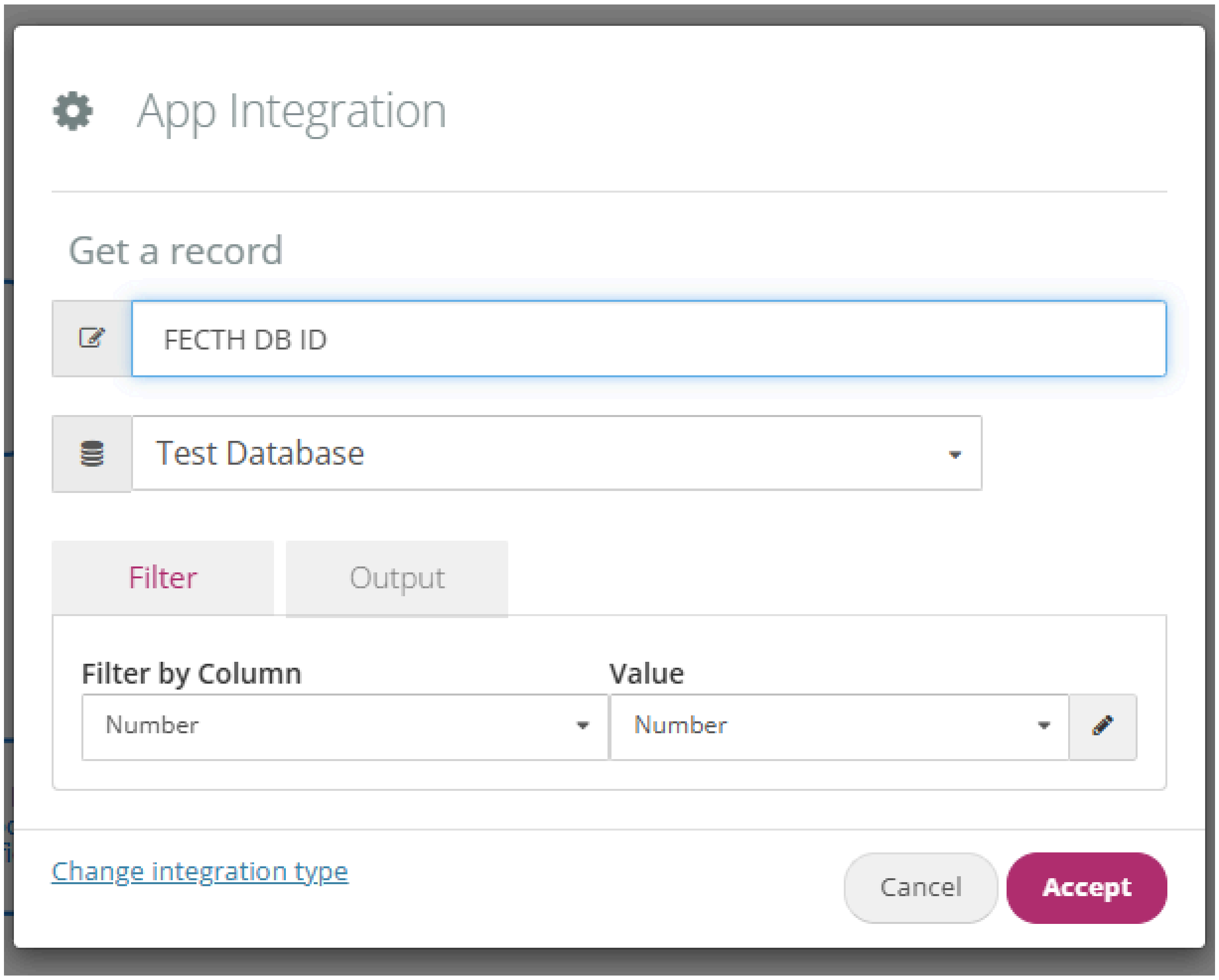
Click here to add a parameter...

[✓] Insert record if it does not exist

Change integration type

Cancel | Accept

Populate Database ID Field in Form



App Integration

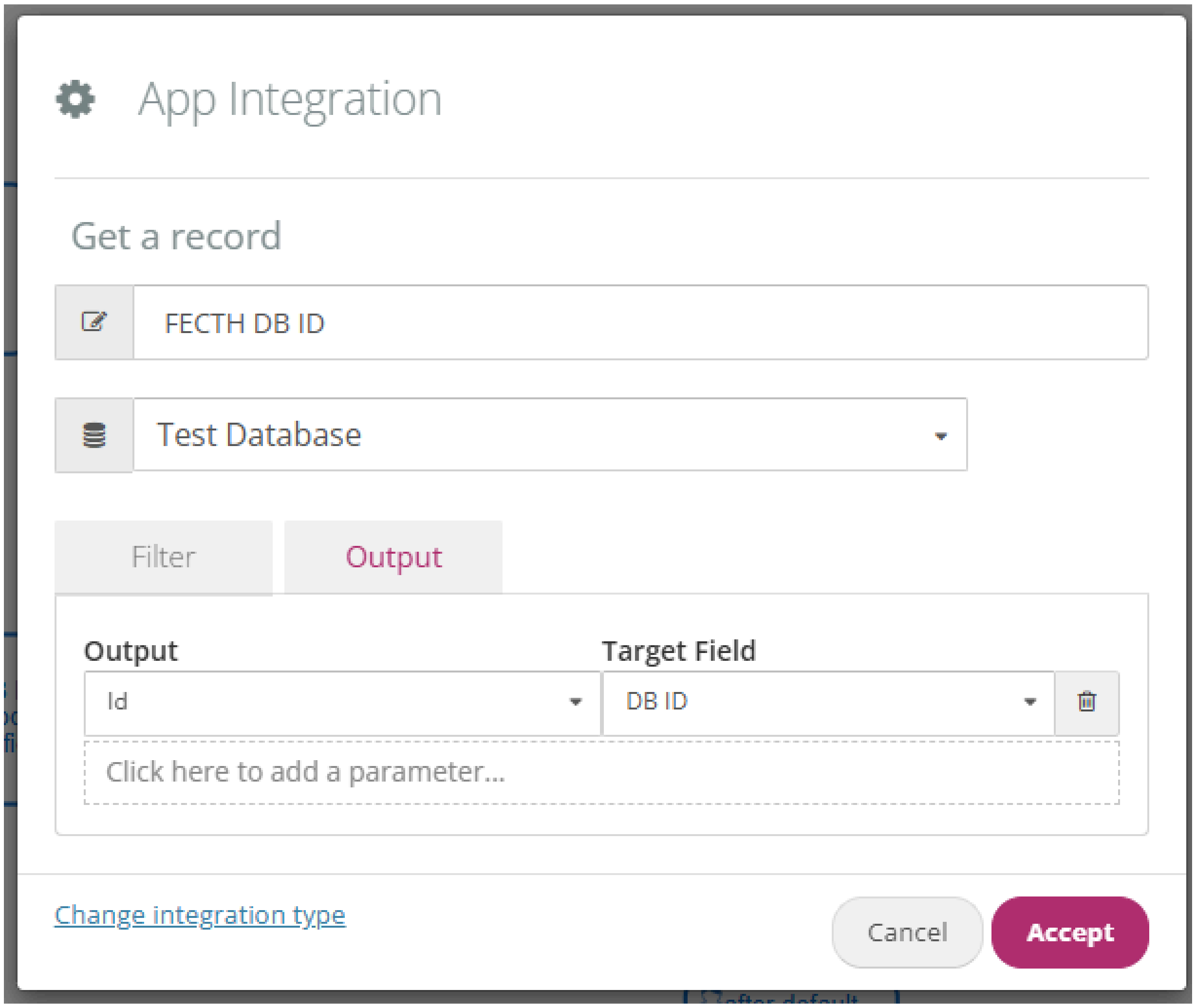
Get a record

FECTH DB ID

Test Database

Filter | Output

Filter by Column	Value
Number	Number


App Integration

Get a record

FECTH DB ID

Test Database

Filter | Output

Output	Target Field
Id	DB ID

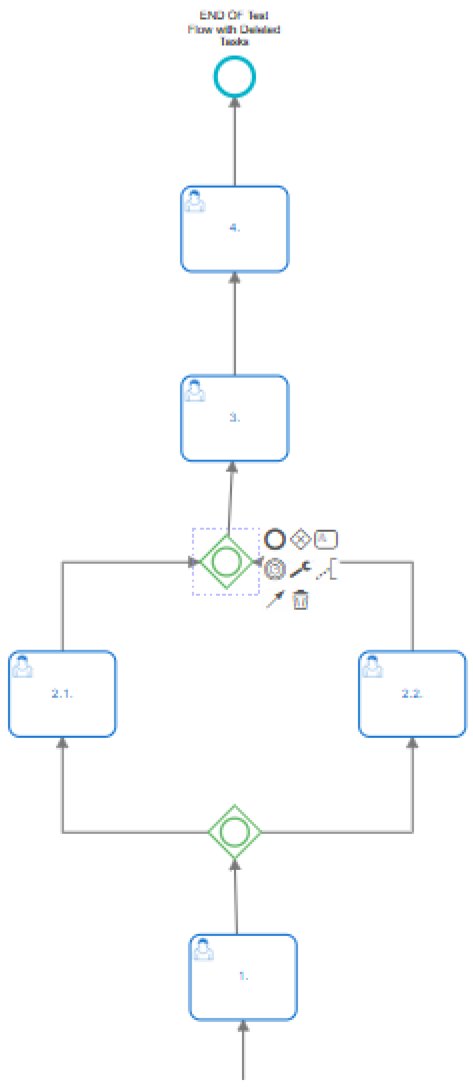
Click here to add a parameter...

Change integration type

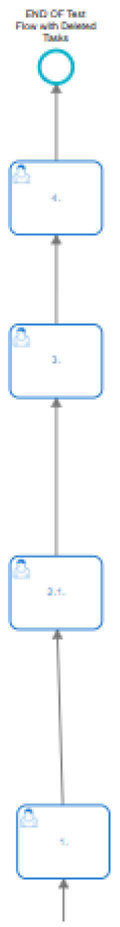
Cancel | Accept

Information: System Behavior When Deleting Tasks in Flows

Before Deletion:



After Deletion: Gateways are also removed.



Tasks Prior to Alteration

Instances in Task 1 follow the updated flow (→ 2.1 → 3). Previously created tasks adopt the newest flow.

Instances in Deleted Tasks

Instances in Task 2.2 (with 1 and 2.1 completed) remain in the inbox and follow the previous flow. Since parallel Task 2.1 is completed and waiting at the inclusive gateway, deletion does not cause errors, even for in-use tasks.

Instances Parallel to Deleted Tasks

Instances in Task 2.1 (with 1 and 2.2 completed) remain in the inbox and follow the updated flow (no longer requiring 2.2 completion before Task 3).

Information: System Behavior with Auto-Completed Tasks Regarding Gateways

Links to Information: System Behavior with New Form Fields Used in Gateways.

Auto-completed tasks save field values from the last update as if completed. Unsubmitted changes propagate upon auto-completion.

Warning: Problematic if users alter gateway-dependent fields (e.g., dropdowns) without updating related fields.

Example: User must complete a date and set dropdown to "Dates Specified" to skip a task. Failure: Dropdown updated without date; task not re-initialized, date unset.

Solutions (Use Exclusively)

Script to Set Required Fields Based on Gateway Field

```
function setRequired() {
    var option = Flokzu.getFieldValue([[Dropdown]]);

    if (option) {
        if (option === "Values Updated") {
            Flokzu.setRequired([[Value 1]]);
            Flokzu.setRequired([[Value 2]]);
        }
    }
}

Flokzu.onChange([[Dropdown]], setRequired);
```

Tip: Place "Dropdown" above other fields to guide updates.

Warning: Timer-completed tasks ignore empty required fields.

Script to Revert Gateway Field if Required Fields Missing

```
function forceValue() {
    var v1 = Flokzu.getFieldValue([[Value 1]]);
    var v2 = Flokzu.getFieldValue([[Value 2]]);
    var option = Flokzu.getFieldValue([[Dropdown]]);

    // Check if either is empty, null, or undefined
    if (!v1 || !v2) {
        // Only reset if the status is not already "Values Not Updated"
        if (option !== "Values Not Updated") {
            Flokzu.setFieldValue([[Dropdown]], "Values Not Updated");
            Flokzu.log("Please add the Value 1 and Value 2 fields before
altering this dropdown");
        }
    }
}

Flokzu.onChange([[Dropdown]], forceValue);
```

Tip: Place "Dropdown" below other fields. Explain in field description (e.g., "Update XYZ first").

Warning: Tasks with required fields auto-complete if blank.

Information: System Behavior with New Form Fields Used in Gateways

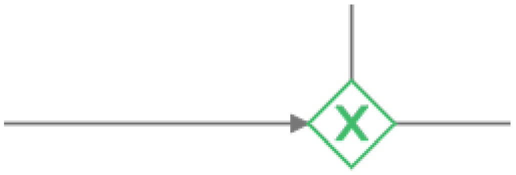
(Tested for inbox tasks and timers.)

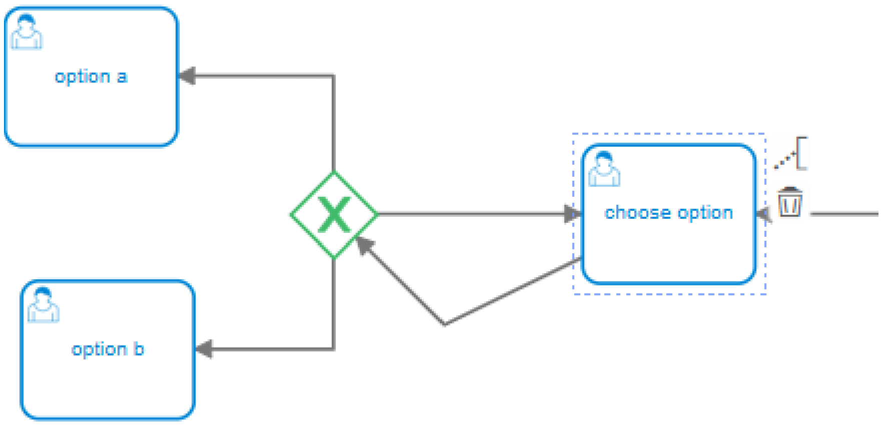
Gateways can use new fields for pre-existing instances, but do not delete prior fields.

New fields appear in existing instances post-update and are usable in gateways. Set defaults directing to flows where the field updates, aligning with expected "default flow."

Warning: No-condition paths in exclusive gateways are followed even if other conditions match.

Notice: Works for linear exclusive gateways (follows met conditions; no-condition if unmet). Fails with recursive loops.





Information: System Handling of Field Name Changes

(Tested for inbox tasks and timers.)

Warning: Manually update scripts.

Note: Field names do not update in timer instances but change upon progression. Reference new names in communications (e.g., `{{Updated Field Name}}`); old names yield blanks.

End of Document
Last updated: November 04, 2025