

# Initiator Conditions and Iteration

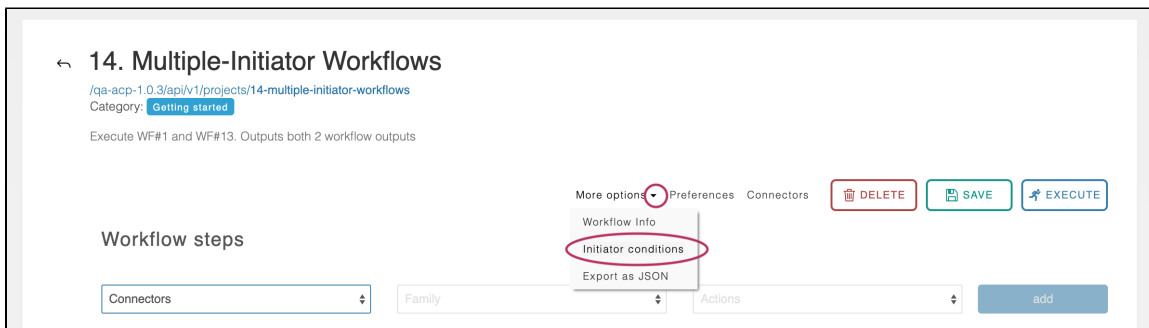
## Initiator Conditions and Iteration

- Initiator Conditions and Iteration
  - Initiator Conditions
  - Iteration Options
  - Workflows Executing Workflows
  - Additional Information
    - ACP User Guide

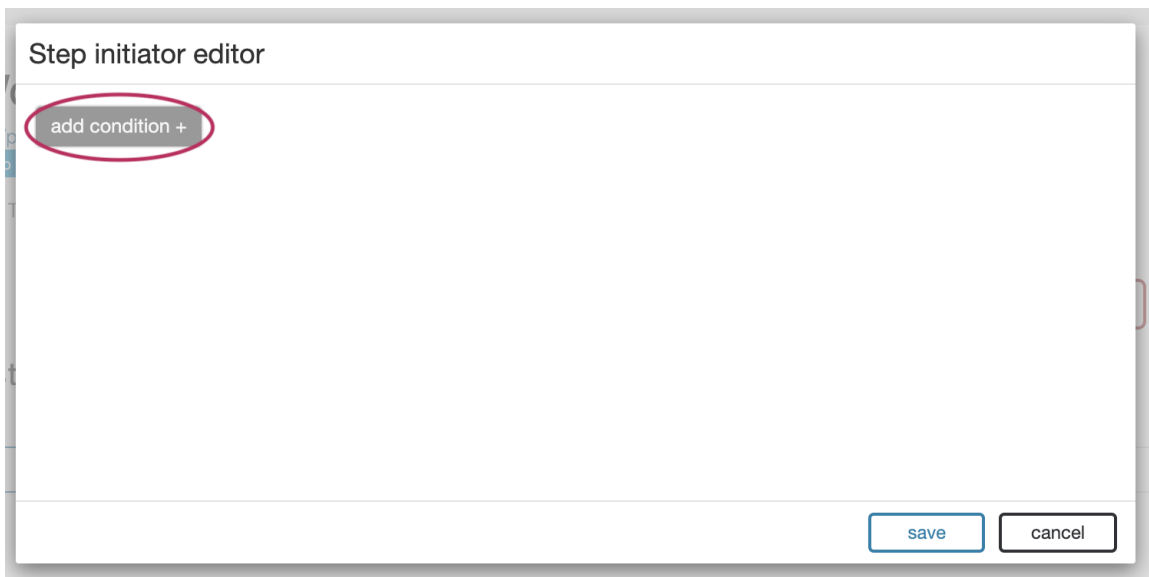
## Initiator Conditions

The 6connect ACP system uses “initiator conditions” as a pre-routing system to determine, based on user-given data, which step is executed first. This is particularly useful for data unification processes. For example, a Workflow can be configured to accept both a Customer Name or a Customer Id, but if a Customer Name is provided to start at a step which first looks up the Customer Id before proceeding to the main work.

The Initiator Editor can be accessed by scrolling to the top of a workflow and clicking ‘More options’, followed by ‘initiator conditions.’



The Initiator Editor consists of a series of conditions blocks identical to those found in the Conditions and Routing sections on each individual step. The “add condition” button can be used to include an arbitrary number of initiator conditions.



Each individual condition can describe the relation between any number of variables using the same tools available in the routing section of individual steps.

The image shows a 'Step initiator editor' window. At the top, there is a button labeled 'add condition +'. Below it, a blue header bar displays 'Condition: success-1' with a trash icon on the right. The main area contains a visual logic builder with labels 'left value', 'operator', and 'right value'. The 'left value' is represented by a large opening curly brace '{', the 'operator' is 'NOT NULL', and the 'right value' is a closing curly brace '}' followed by a plus sign in a circle '+'. Below this, there is a 'Step:' label followed by a dropdown menu. At the bottom right, there are 'save' and 'cancel' buttons.

When the workflow is executed each condition is evaluated against the user-given data, from top to bottom, in order. Once a matching condition is found the Workflow immediately proceeds to the indicated step.

## Iteration Options

Each Step has an “Iteration Options” section which can be used to execute a single step multiple times.

The image shows the 'Iteration Options' tab of a workflow step configuration. The step is named 'PUT Smart assign'. The 'Iteration variable' field has a dropdown menu open with the following options: 'default value', 'user given' (which is selected with a checkmark), 'workflow link', 'function', and 'omitted'. To the right of the dropdown is a 'variable name...' input field. The background shows other tabs like 'General Options', 'Required Inputs', 'Optional Inputs', 'Sub Steps', 'Conditions & Routes', and 'Output'.

Each iteration cycle will execute the step with identical inputs, with the exception of any variables defined with the ‘iteration’ type. These variables will be supplied with data derived from the specific iteration cycle.

The Iteration Variable field in the Iteration Options section accepts the normal set of ACP variable types. If supplied with a simple numerical value (ex: 5), the step will iterate that number of times. If supplied with a JSON array (ex: [ { id: 123}, { id : 321 } ] ), the step will repeat once for every member of the array.

Once a Step’s Iteration Variable is changed from “omitted,” each Input across all sections gains access to the “iteration” data type. If selected, this input will be supplied with data from the iteration cycle.

In the case of simple numeric iteration, the data supplied will the iteration count, starting from zero.

In the case of a more complicated JSON structure (ex: [ { id: 123}, {id : 321 } ] ), the entire object will be provided to the variable. If only part of the structure is required, it can be accessed using the same semantics used to navigate JSON trees elsewhere. In the above example “{id: 123}” will be passed to the input by default, but if the input is configured with “id”, then only the value “123” will be selected. Likewise, if the object contained an array, then “2.id” will select the third item’s “id” parameter.

## Workflows Executing Workflows

ACP ships with a self-referential “acp” connector type which is designed to include API calls generated by the ACP system itself. This connector type can be found from the general list of connector types on the Connectors modal.

This functionality is extremely useful as it allows the user to extend simple workflows into more advanced ones in an intuitive fashion.

For example, imagine a situation where a Workflow integrates between some 3rd party billing software and customer information stored in 6connect ProVision. A simple workflow might accept user credentials (user names, account numbers, etc) and then verify them with ProVision, the billing software, or both. This simple workflow would return the user object on success, but an error on failure.

On its own, this doesn't seem to do much, but this Workflow can be used as the first step of all subsequent Workflows, drastically cutting down on the overhead as well as allowing for easy single-point for maintenance in case of shifts in the underlying technologies.

By creating simple workflows for very focused tasks one can build up a library of increasingly more complex combinations, allowing for very powerful results to be achieved in a compact space.

## **Additional Information**

Continue on to additional User Guide pages for detailed information on working in ACP:

### **ACP User Guide**

- [User Management](#)
- [Workflow Overview](#)
- [Connectors](#)
- [Workflow Steps](#)
- [Workflow Options](#)
- [Executing Workflows and Export](#)
- [Initiator Conditions and Iteration](#)
- [Javascript Functions](#)
- [Template Workflows](#)