Branching Logic

CCTS Biostatistics Core

October 2024

What is Branching Logic?

Copied from REDCap Help and FAQ:

Branching Logic may be employed when fields in the database need to be hidden during certain circumstances. For instance, it may be best to hide fields related to pregnancy if the subject in the database is male. If you wish to make a field visible ONLY when the values of other fields meet certain conditions (and keep it invisible otherwise), you may provide these conditions in the Branching Logic section in the Online Designer (shown by the double green arrow icon), or the Branching Logic column in the Data Dictionary.

For basic branching, you can simply drag and drop field names as needed in the Branching Logic dialog box in the Online Designer. If your branching logic is more complex, or if you are working in the Data Dictionary, you will create equations using the syntax described below.

In the equation you must use the project variable names surrounded by **[ ]** brackets. You may use mathematical operators (=,< ,>,<=,>=,<>), Boolean logic (and/or), and unary Boolean not (!). You may nest within many parenthetical levels for more complex logic.

You must **ALWAYS** put single or double quotes around the values in the equation UNLESS you are using > or < with numerical values.

The field for which you are constructing the Branching Logic will ONLY be displayed when its equation has been evaluated as TRUE. Please note that for items that are coded numerically, such as dropdowns and radio buttons, you will need to provide the coded numerical value in the equation (rather than the displayed text label). See the examples below.

[sex] = "0"	display question if sex = female; Female is coded as 0, Female
[sex] = "0" and [given_birth] = "1"	display question if sex = female and given birth = yes; Yes is coded as 1, Yes
([height] >= 170 or [weight] < 65) and [sex] = "1"	display question if (height is greater than or equal to 170 OR weight is less than 65) AND sex = male; Male is coded as 1, Male
[last_name] <> ""	display question if last name is not null (aka if last name field has data)

Figure 1: REDCap Help and FAQ - Branching Logic

Example

You want to ask a survey participant to specify their ethnicity when 'Other' (option code 9) is selected for the multiple-choice variable `ethnicity`. Add the branching logic to the following field, `specify_ethncity`:

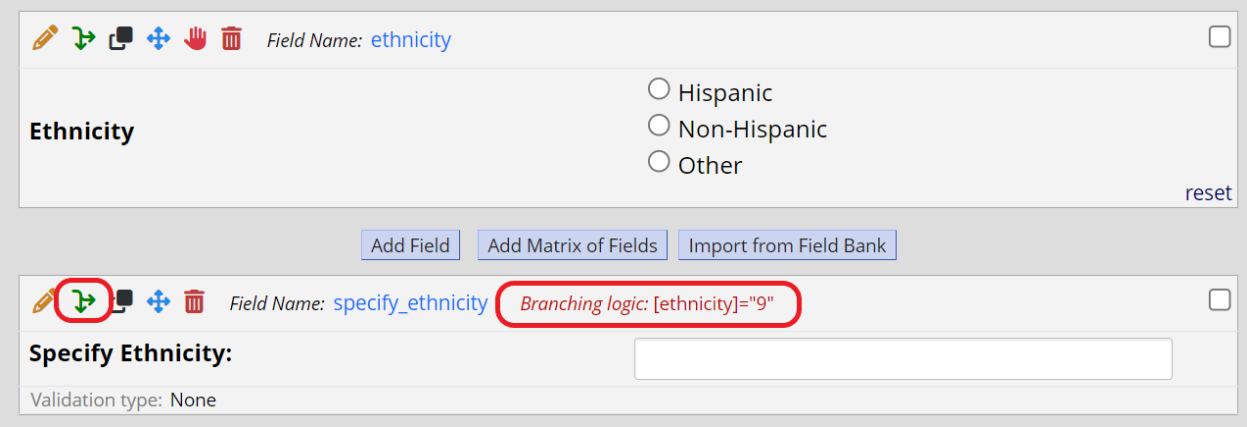


Figure 2: Branching Example

Instructions

Click the green arrow icon above a question to open the branching logic editor. Remember to apply the logic to the field that should be conditionally hidden or displayed.

Use **Advanced Branching Logic Syntax** or the **Drag-N-Drop Logic Builder** to add logic.

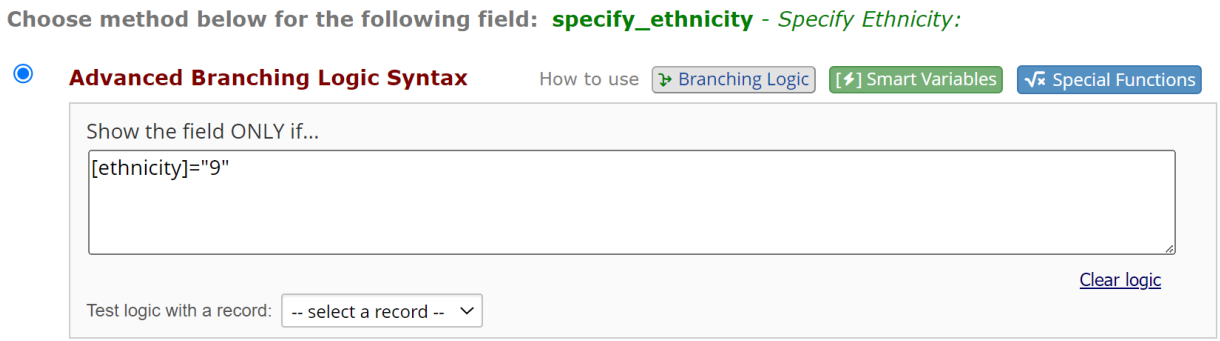


Figure 3: Advanced Branching Logic Syntax

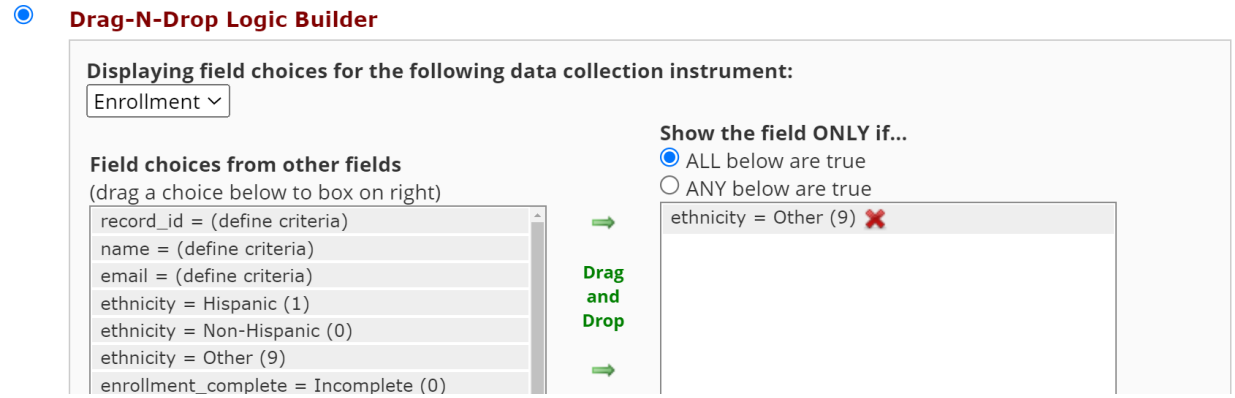
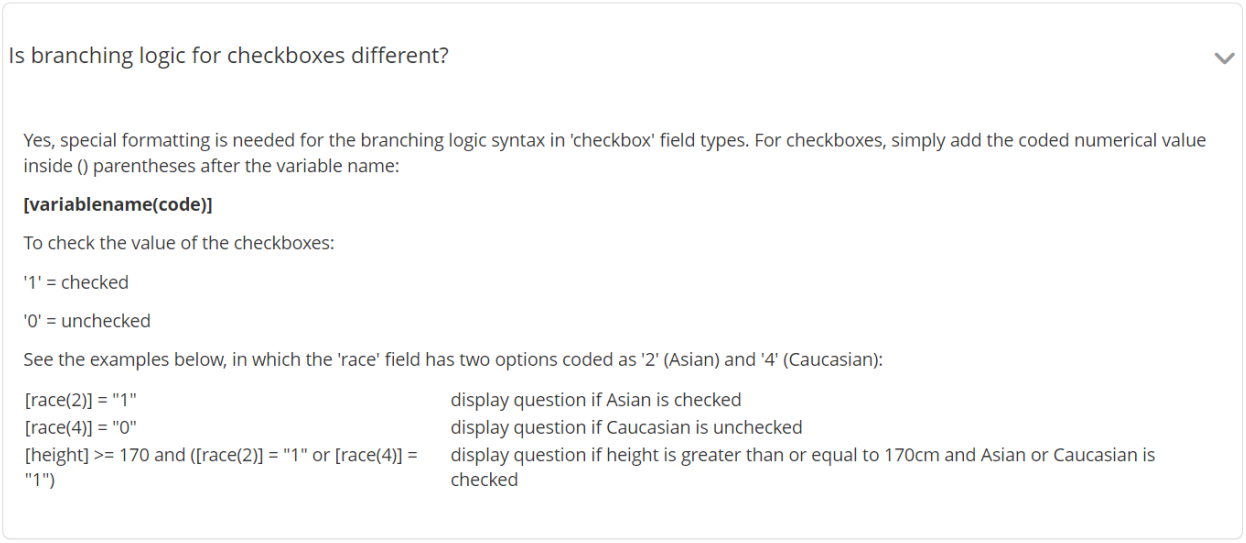


Figure 4: Drag-N-Drop Builder

Logic Based on Checkboxes

Branching logic based on responses to checkboxes uses special syntax. From the REDCap Help and FAQ:



Is branching logic for checkboxes different?

Yes, special formatting is needed for the branching logic syntax in 'checkbox' field types. For checkboxes, simply add the coded numerical value inside () parentheses after the variable name:

[variablename(code)]

To check the value of the checkboxes:

'1' = checked

'0' = unchecked

See the examples below, in which the 'race' field has two options coded as '2' (Asian) and '4' (Caucasian):

[race(2)] = "1"	display question if Asian is checked
[race(4)] = "0"	display question if Caucasian is unchecked
[height] >= 170 and ([race(2)] = "1" or [race(4)] = "1")	display question if height is greater than or equal to 170cm and Asian or Caucasian is checked

Figure 5: REDCap Help and FAQ - Branching for Checkboxes

Longitudinal Projects with Named Events

When a form is repeated in multiple events in a longitudinal project, branching logic refers to values in the **current event** by default.

To use data entered in a different event, you must include the unique event name in brackets before the variable name, in the format **[unique_event_name][variable_name]**. For example, to base logic on a participant's age at baseline, you might use **[baseline_arm_1][age]**.

You can also use a smart variable to refer to another event contextually. For example, the smart variable **[previous-event-name]** will find the value entered in the previous event, as in **[previous-event-name][variable_name]**.

Read more about longitudinal data collection.

Repeating Forms

Copied from REDCap Help and FAQ:

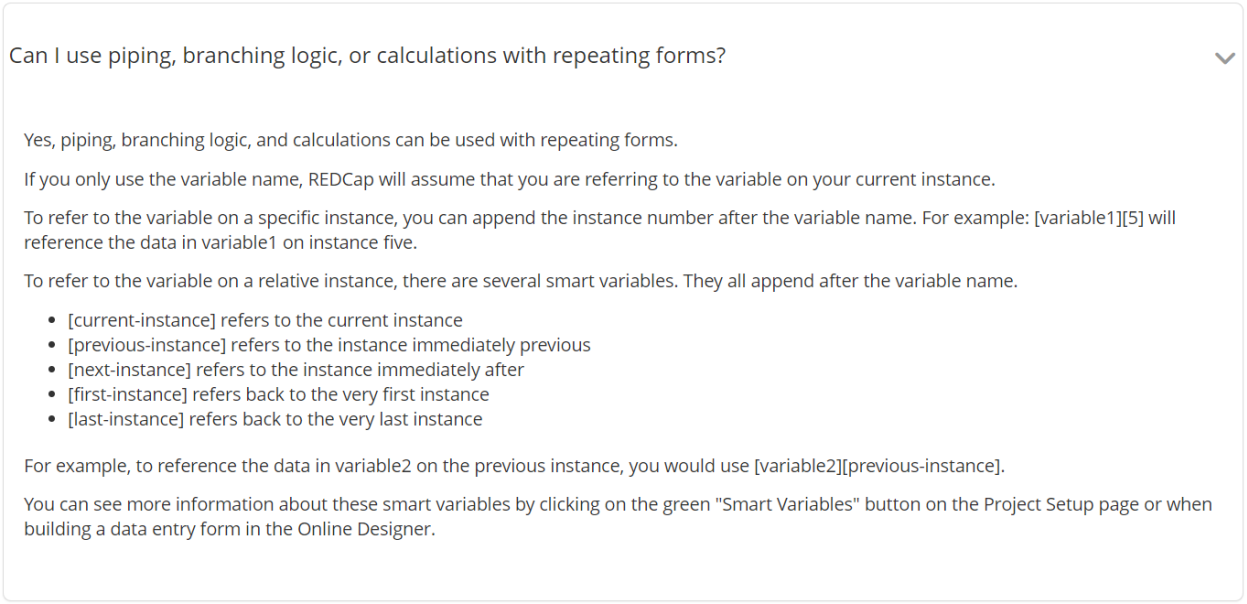


Figure 6: REDCap Help and FAQ - Repeating Forms

Skipping Sections

Branching logic is defined on a field-by-field basis. Although you can create sections using section header fields, you cannot apply branching logic directly to a section or a section header.

However, you can conditionally skip an entire section on a survey if you apply the same logic to **all fields within in the section** (i.e., all fields between two section header fields). If branching logic hides all fields in a section, the section's header will also be hidden. If your survey is set up to display one section per page, the entire page will be skipped. To quickly apply the same branching logic to multiple fields, select them in the Online Designer and click the branching icon in the "Quick-modify field(s)" menu.

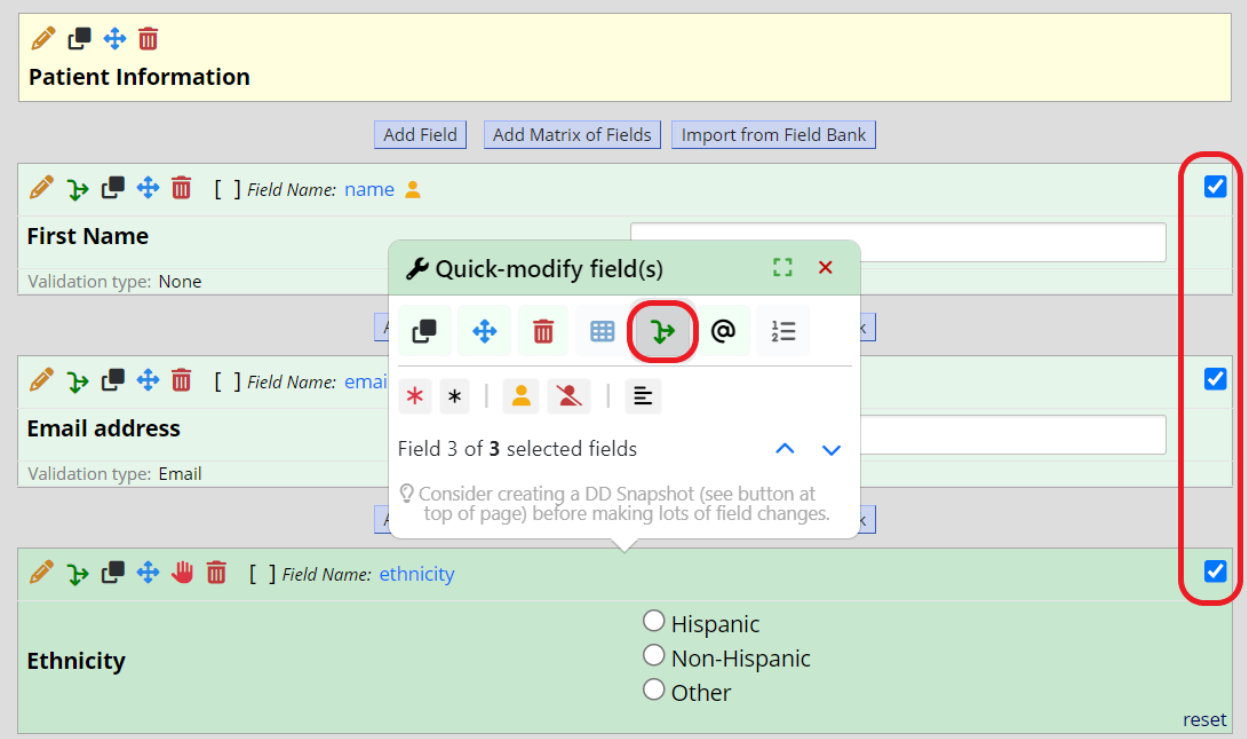


Figure 7: Quick-modify field(s)

Skipping Entire Instruments

If a survey section can be moved to a standalone instrument, consider using **Form Display Logic** for form-level branching. Form Display Logic can disable an entire instrument for data collection until certain conditions are met. The form will still appear on the record status dashboard, but the form will be inaccessible and greyed-out (unless you hide the form using a custom record status dashboard).You can also skip entire surveys by using:

- The **Survey Queue**, found in the Online Designer; or
- **Conditional logic for Survey Auto-Continue**, found in each individual instrument's survey settings

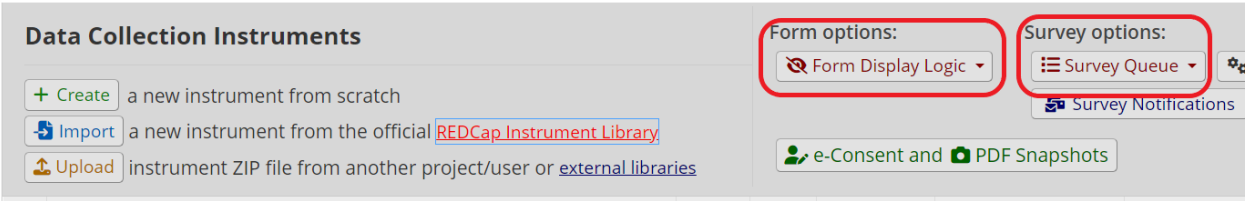


Figure 8: Form Display Logic and Survey Queue

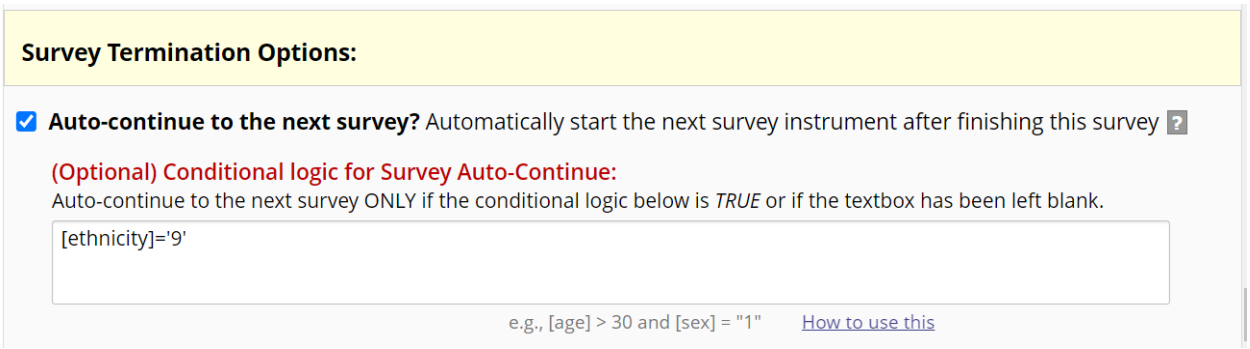


Figure 9: Conditional logic for Survey Auto-Continue

Test Branching Logic

The Online Designer preview won't evaluate branching logic. Instead, test your branching logic on a data entry form or survey by entering test data while in development mode.

If you encounter a problem while entering test data in a form, REDCap will display a popup message. Common issues include:

- Logic syntax issues: Non-existing variable names or invalid conditions
- Data errors: Data exists in fields that should be hidden by new branching logic conditions

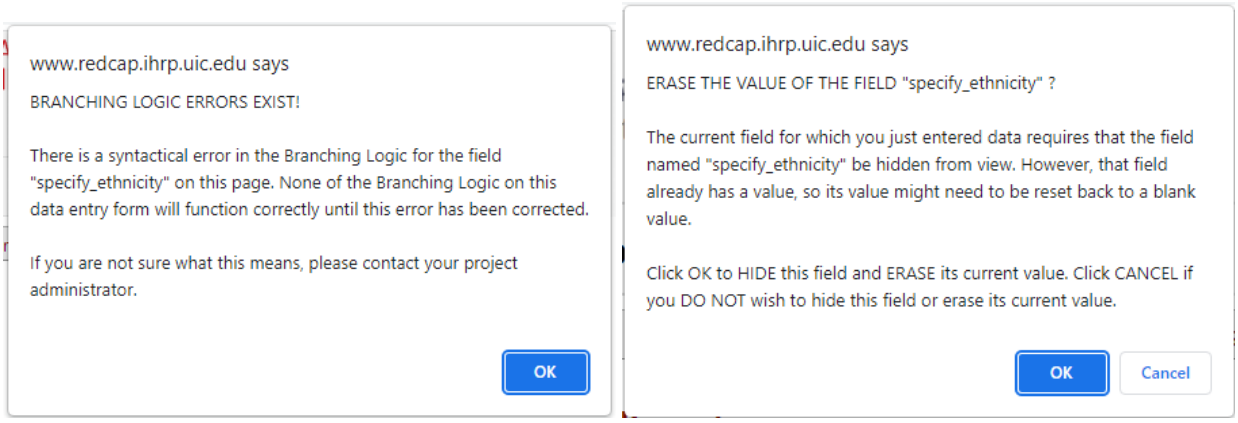


Figure 10: Branching Errors

Branching Logic After Field Modification

If an instrument includes branching logic, carefully review it every time modifications are made, as some field edits can affect the branching logic. If your project is in production status (**highly recommended** when collecting real data), you will have the chance to review the effects of draft modifications.