SKIP FORM LOGIC GUIDE

OVERVIEW

Click Boarding allows full admin users to configure "Skip Form Logic" rules to provide efficient, candidate-specific workflow experiences. For example, a set of candidates might all complete the same process workflow but require different forms within the state wage form step. On this step, Skip Form Logic rules can be utilized to present each candidate with just their own state's form, alleviating the need to create a workflow per state.

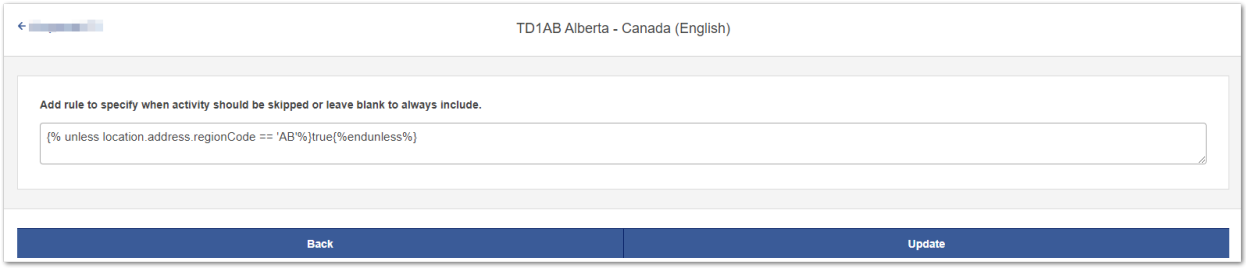
Skip Form Logic is written using a language called "Liquid" and will reference Click Boarding data elements such as those from the candidate and candidate's process location. This guide includes an overview for how the logic works, and an appendix containing examples to provide you with a quick start.

Before You Begin:

- When the rule evaluates to **true** the form is skipped, and when it evaluates to **false** the form is shown.
    - Example: if you have a Minnesota wage form that you only want to show to candidates who work in Minnesota, you can evaluate the candidate's work address state. If it is not MN, then you can tell it to be "true" (skip), else if it is MN, this will evaluate to "false" and show the form.
- Skip Form Logic can only be used on a data field that existed on a candidate at the time their process workflow was assigned. If a candidate enters data on a form mid-process, that data cannot be used in Skip Form Logic within the same process flow because it was added after the process was assigned.
- Skip form logic can only be used on data that resides within certain candidate objects. This includes the candidate, location, and process. See Appendix B in this guide.

WHERE TO CONFIGURE

Users who can edit candidate processes within the Click Boarding Admin Portal can set up skip form logic at the *form level* within a process step (click *Edit* on a form in a step):

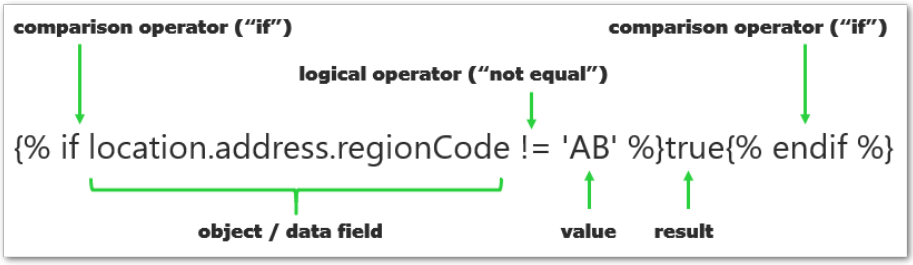


If the Skip Form Logic box is left empty, the form will always be shown ("false"), which is the default.

LIQUID TEMPLATE LANGUAGE

Liquid is an open source language that provides a safe and flexible way to write logic statements and is used in many different popular platforms and websites. Because the syntax can look a bit technical to first-time users, we have provided a basic tutorial and example set so you can copy, paste, and edit as needed without having to start from scratch.

Example



This liquid expression says if the candidate's process location region code isn't Alberta, skip the form. The entire expression is a "comparison statement" starting with "if" and ending with "endif." We are comparing a field (regionCode) to an expected value (AB) with some logic (not equals) and outputting a result (true). Implied in this logic is if we do get AB, then the statement is false—which can be written into the expression, but it is not required. Then the "endif" says we are done. We will elaborate more below.

Tags

Each rule is encapsulated by "tags" made of curly braces and percent signs: **{%** and **%}**. These control the flow of the expression to encapsulate and "hide" the logic. The parts of the expression that are not inside the tags are the *results*, i.e., whether the expression is **true** or **false**. Here are the tags from the example above:

Fields

An important prerequisite to building Skip Logic rules is to know the field name(s) that you are using. The field name is what comes back in the API when the candidate, location, or process is called, and is visible in the admin portal "Data Fields" section. This name can also be provided through your support channel if you can't find the name of the field that you are using in a form. Some fields in Click Boarding are considered standard, because they are part of the base candidate profile: name, email, phone, address, etc. Other fields may be custom fields that you have created specifically for your company.

When referencing a field in Click Boarding for Skip Logic, we use dot notation where the name consists of the full path to the field, and periods (dots) are placed between each element representing the location of the field within its object (candidate, process, or location) and data field group. For example, a custom candidate field called "my_field_name" would be written like this:

As far as Liquid is concerned, the entire thing is an "object." But to be more specific in Click Boarding, candidate is the object, customFields is a data field group, and my_field_name is the field name itself.

This path is visible both within the API, and within the Data Fields section of the Admin Portal. Custom fields are most often stored under the **candidate.customFields** section, unless you have defined custom fields differently via data field creation in the admin portal. See Appendix B, "Data Context Example," for more information on field names & structure.

A few more field path examples:

Candidate home address postal code	**candidate.addresses.home.postalCode**
Candidate home state	**candidate.addresses.home.regionCode**
Candidate work state	**candidate.addresses.work.regionCode**
CandidateProcess Location name	**location.name**
CandidateProcess Location city	**location.address.city**

Logic/Operators

To control the logic in each skip form rule, the following operators can be used. We will include some specific examples later to illustrate. Note: when multiple operators are used, the order of operations is from right to left and parentheses cannot be used to alter that order.

Logical operator	Logical operator end	Description
==		Equals
!=		Does not equal
>		Greater than
<		Less than
>=		Greater than or equal to
<=		Less than or equal to
or		Logical or
and		Logical and
if	endif	Conditional
unless	endunless	Conditional

True/False and Empty/Null

Logic must evaluate to either true or false. If a data field contains a string (any text) or is empty (existed at one point but was blanked out) it will evaluate to **true**, and if nil/null (missing/never existed), it will evaluate to **false**. For example:

```
{% if candidate.customFields.skip_screening %}true{% else %}false{% endif %}
```

In this rule, if the "skip_screening" field is populated with any value, the rule will evaluate to "true" and skip the form. If the field is null/missing, then the form will be shown.

It is technically OK to write a "false" statement, but note that the system's default is also "false" so it will not automatically default the comparison result to "true" unless you specifically write it in. When writing a rule that evaluates your primary case to "false," you must add the rest of the rule to avoid unexpected results: "… {% else %}true{% endif %}"

Examples

To combine what we have outlined so far and illustrate how this comes together, take the following example:

```
{% if candidate.customFields.remote == 'Yes' or candidate.customFields.sales == 'Yes' %}true{% endif %}
```

This rule that says if a candidate's custom field for "remote" or "sales" contains the value "Yes", then skip the form ("true"). It's an example of how you can use an operator like "or" within the "if" condition to look at more than one field, and you can specify what the *value* in either of those fields must be in order to be true. Similarly, you could use the "and" operator to require both fields to be "Yes" in order to skip the form, like this:

```
{% if candidate.customFields.remote == 'Yes' and candidate.customFields.sales == 'Yes' %}true{% endif %}
```

The example below is slightly more complex but is still simple to write and will save time. In this scenario we want to only show a form to California residents. Rather than writing a long rule that contains *every other* US state and territory so any of those can evaluate to true, we could flip the rule around so if we see California the rule evaluates to false, and any other value that *isn't* California will evaluate to true:

```
{% if candidate.addresses.home.regionCode == 'CA' or location.address.regionCode == 'CA' %}false{% else %}true{% endif %}
```

But it's better to have the rule evaluate to true first. There is no default that will turn an "else" case into "true" so if you don't write else/true into your rule, everyone would see the form because we default the else case to false. To that end, here's another way to write this:

```
{% unless candidate.addresses.home.regionCode == 'CA' or location.address.regionCode == 'CA' %}true{% endunless %}
```

APPENDIX A: EXAMPLES

Below is a table containing some common examples for Skip Logic that our customers use.

Description	**Liquid expression for Skip Logic**
Skip this form if candidate field called “seasonal” has been populated.	{% if candidate.customFields.seasonal %}true{% endif %}
Skip this form if the candidate’s home state isn’t Texas.	{% if candidate.addresses.home.regionCode != 'TX' %}true{% endif %}
Unless the candidate’s home state or process location state is California, skip this form.	{% unless candidate.addresses.home.regionCode == ‘CA’ or location.address.regionCode == 'CA' %}true{% endunless %}
Skip this form if the candidate’s location region code isn’t Alberta. This example could be used for all Canadian TD1 forms, swapping out each province code as noted here.	{% unless location.address.regionCode == 'AB' %}true{% endunless %} CA provinces: AB, BC, MB, NB, NL, NS, NT, NU, ON, PE, SK, YT
Another version of the above that adds the candidate’s home region.	{% unless candidate.addresses.home.regionCode == ‘AB’ or location.address.regionCode == ‘AB’ %}true{% endunless %}
Skip this form if the candidate’s location identifier is HQ001.	{% if location.identifier == 'HQ001' %}true{% else %}false{% endif %}
Skip this form if the field part_time doesn’t contain the value “Yes.”	{% if candidate.customFields.part_time != ‘Yes’ %}true{% else %}false{% endif %}
Unless field “part_time” and field “seasonal” are both “Yes,” skip the form.	{% unless candidate.customFields.part_time == ‘Yes’ and candidate.customFields.seasonal == 'Yes' %}true{% endunless %}

APPENDIX B: DATA CONTEXT EXAMPLE

Skip Form Logic has a certain data context that can be used to drive its logic. “Data context” refers to a relevant set of objects that contain fields that are useful to this feature. For example, Brands and Tasks are not part of the data context for Skip Form Logic, but Candidate, Candidate Process, and Location are part of its data context.

Below is an example of the data context as it would look in the API, showing the field hierarchy. Candidate fields will vary for each client. The highlights show how you can find the path for a field like *candidate.addresses.home.city*.

```
{
    "candidate": {  // See Candidate Api
        "id": "x3rh9s",
        "name": {
            "first": "John",
```

```
    "last": "Doe"
},
"email": {
    "address": "jdoe@clickboardingdemo.com"
},
"phones": [{
    "number": "1112223333",
    "type": "Home"
}],
"addresses": {
    "home": {
        "type": "Home",
        "streetLine1": "123 Main St",
        "city": "Eden Prairie",
        "countryCode": "US",
        "regionCode": "MN",
        "postalCode": "55347"
    }
},
"dateOfBirth": "1970-01-01",
"directDeposit": {
    "optIn": false
},
"eeoc": {
    "sex": "Male"
},
"emergencyContacts": {
    "optIn": false
},
"ssn": "111221111",
"customFields": {
    "MyDesk_IdealWorkspace": "Yes",
    "MyDesk_ChooseYourLaptop": "Macbook",
    "MyDesk_ChooseMonitorSetUp": "Dual Monitor",
```

```
            "MyDesk_ChoosePhone": "iPhone",

            "MyDesk_ChooseSoftware": "Yes"

        }

    },

    "location": { // See Location Api

        "name": "Box Car Billy's",

        "id": "OhZCjn_-D0mfE4nWrpnupA",

        "identifier": "BoxCarBillys",

        "address": {

            "streetLine1": "11100 Wayzata Blvd",

            "streetLine2": "",

            "city": "Minnetonka",

            "regionCode": "MN", // regionCode is populated when countryCode is US, CA, or AU. Not
available at the same time as regionName

            "regionName": "BAL", // regionName will be populated when regionCode is not. Not
available at the same time as regionCode

            "postalCode": "55305",

            "countryCode": "US"

        },

        "contactUser": {

            "id" : "a1b2c3",

            "active" : true,

            "createdOn" : "2019-04-16T18:34:10Z",

            "email" : "locationContact@example.com",

            "name" : {

                "first":"ContactFirst",

                "last":"ContactLast"

            }

        },

        "settings": {

            "eVerifyRequired": true

        },

        "brandLogo" :{

            "contentType": "image/png",

            "content": "iVBORw0KGgoAAAANSUhEU...snip...UVORK5CYII="
```

```
    }
},
"process": { // See CandidateProcesses Api
    "dueDate": "2019-11-14T22:07:22Z",
    "candidateDueDate": "2019-11-07T22:22:17Z",
    "process": {
        "name": "Process Flow Name",
        "referenceName": "Unique Process Flow Identifier"
    },
    "roles": {
        "hiringManager": {
            "userId": "v7npyq0",
            "id": "v7npyq0",
            "createdOn": "2019-01-01T16:35:14Z",
            "active": true,
            "name": {
                "first": "HiringManager First Name",
                "last": "HiringManager Last Name"
            },
            "email": "hiringManager@domain.com"
        },
        "coordinator": {
            "userId": "mrpdkqy",
            "id": "mrpdkqy",
            "createdOn": "2019-01-01T17:14:04Z",
            "active": true,
            "name": {
                "first": "Coordinator First Name",
                "last": "Coordinator Last Name"
            },
            "email": "coordinator@domain.com"
        },
        "recruiter": {
            "userId": "e026wdg",
```

```
                "id": "e026wdg",
                "createdOn": "2019-01-31T18:05:43Z",
                "active": true,
                "name": {
                    "first": "Recruiter First Name",
                    "last": "Recruiter Last Name"
                },
                "email": "recruiter@domain.com"
            }
        }
    }
}
```