

Create a Type Ahead Block Element



NOTE:

Did you sign up for a [special OmniStudio Developer Edition org](#) already? You'll need one to do the steps in this guide. If not, use the link to fill out the form and have an org delivered to your inbox. The Exercise Guide in the first unit of this module has more detailed steps for this process if you need them.

Requirements

“As a service agent, I'd like a more complex guided workflow that provides me with an option to change the primary contact to another existing contact that I easily lookup (and edit their contact information)”

Based on the user requirements, you add a Type Ahead Block to the BlkChangePriContact element of your branching OmniScript. A Type Ahead Block is an auto-complete feature. Once configured, it retrieves data and displays it in a dropdown list as the user types, saving them from having to type the full value.

Prerequisites

- Build an OmniScript with Branching

Tasks

1. Add a Type Ahead Block
2. Refine the UI of the Type Ahead Block

Time

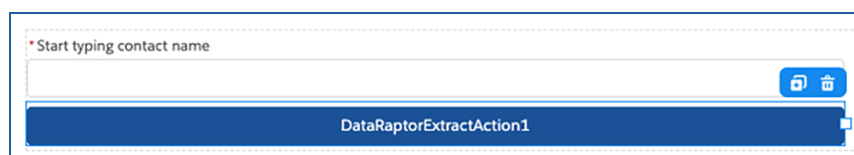
- 35 mins

Task 1: Add a Type Ahead Block

1. Add a Type Ahead block to the **BlkChangePriContact** block.
 - a. In the header, click **Design**.
 - b. From the **Build** panel, expand the **GROUPS** section or use **Search** and enter `type` and locate the **Type Ahead Block** element.
 - c. Drag the **Type Ahead Block** into the **BlkChangePriContact** block on the Canvas (the empty block).
 - d. In the **Properties** panel, set the following values:

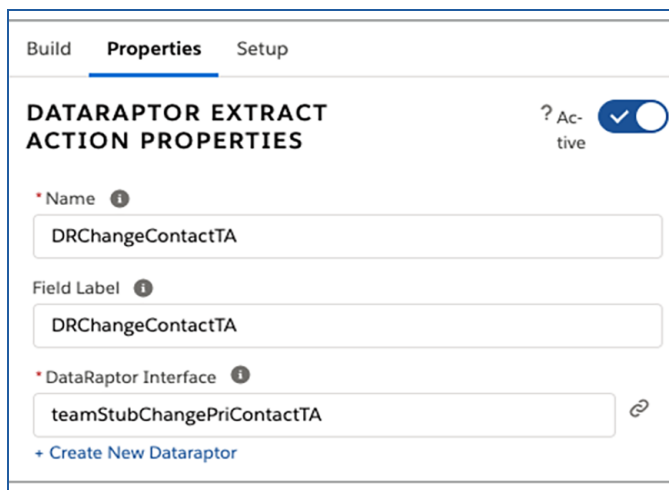
| Property | Value | Notes |
|-------------|---------------------------|--|
| Name | TChangeContact | TA is an abbreviation for “typeahead”. |
| Field Label | Start typing contact name | User instructions |
| Required | ✓ | |

2. Configure a data source for the Type Ahead block. A stub DataRaptor Extract is already available for you to use.
 - a. In the **Build** panel, expand the **ACTIONS** group (if you used search, remember to clear the search conditions).
 - b. Drag a **DataRaptor Extract Action** into the **TChangeContact** block. (It appears as a button inside the block.)



- c. For **Name** and **Field Label**, change the value to `DRChangeContactTA`.

- d. From the **DataRaptor Interface** dropdown list, select **teamStubChangePriContactTA**.



The screenshot shows the 'Properties' tab of a 'DATARAPTOR EXTRACT ACTION' configuration. It includes a toggle switch for 'Active' (checked), a 'Name' field with the value 'DRChangeContactTA', a 'Field Label' field with the value 'DRChangeContactTA', and a 'DataRaptor Interface' dropdown menu with the value 'teamStubChangePriContactTA'. A link '+ Create New Dataraptor' is visible at the bottom.

3. Add input parameters.

As with an Integration Procedure, you need to add some input parameters to send data to the DataRaptor. This DataRaptor only needs to look for contacts associated with this account.

Input parameters consist of a data source and a filter value. The data source specifies the OmniScript element or JSON node that is passed to the DataRaptor, and the filter value renames it to a name that the DataRaptor expects. In this example, the first input parameter passes the AccountId to the DataRaptor to filter the search to the account's contacts. The second input parameter sends the text that the user types in to the DataRaptor as the lookup key.

- a. Set the following values:

| Data Source | Filter Value |
|-------------|--------------|
| ContextId | AccountId |

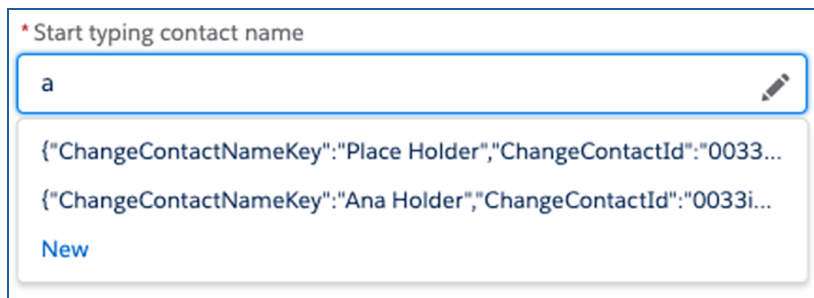
- b. Click **+ Add Input Parameter**.

- c. Set the following values:

| Data Source | Filter Value |
|----------------|--------------|
| TChangeContact | LookupKey |

The first value limits the search to the contacts already associated with this account. The second sends what the user types into the Type Ahead field. The first values have to be typed in, which the second Data Source selected from the list of options in the dropdown list.

4. Review the data flow updates in the UI.
 - a. In the Header, click **Preview**.
 - b. Select the **Change primary contact** radio button.
 - c. Type **a** in the field.
 - d. Notice the dropdown list has all of the Contact stub data. You need to finish configuring the Type Ahead Block to only display Contact names that match what is entered.

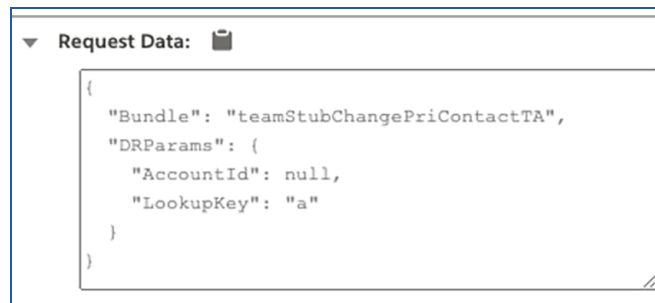


* Start typing contact name

a

{ "ChangeContactNameKey": "Place Holder", "ChangeContactId": "0033..."
{ "ChangeContactNameKey": "Ana Holder", "ChangeContactId": "0033i..."
New

- e. Click **Action Debugger**.
- f. Expand the most recent entry **DRChangeContactTA**. You may need to scroll down.
- g. Expand the **Request Data** node (you may also have to expand the box by dragging the lower-right corner to see the entire node).



- h. Notice the OmniScript is sending a **LookupKey** and an **AccountId** to the DataRaptor. (Because you are using stub data, there is no AccountId being sent.)
 - i. Expand the **Response** node under the request data node to view the response from the DataRaptor.
 - j. Notice the response contains two contact records with the following field names:
 - i. ChangeContactNameKey
 - ii. ChangeContactId
 - iii. ChangeContactEmail
5. Configure the name of the field to display in the Type Ahead Block.
- a. From the first contact record, copy the text **ChangeContactNameKey** to the clipboard.

```
Apex Class: vlocity_ins.DefaultDROmniScriptIntegration

Apex Method: invokeOutboundDR

▶ Remote Options: 📁

▶ Request Data: 📁

▼ Response: 📁

{
  "OBDResp": [
    {
      "ChangeContactNameKey": "Place Holder",
      "ChangeContactId": "0033i0000029WovZZZ",
      "ChangeContactEmail":
        "pholder@placeholderinc.com"
    },
    {
      "ChangeContactNameKey": "Ana Holder",
      "ChangeContactId": "0033i0000029Wp0ZZZ",
      "ChangeContactEmail":
        "aholder@placeholderinc.com"
    }
  ]
}
```

- b. In the header, click **Design**.
- c. In the Canvas, click the **TChangeContact** element.
- d. In the **Properties** panel, locate **Typeahead Key** and paste the JSON key from the clipboard.
- e. Return to **Preview**, select **Change primary contact**, and type a.
- f. The dropdown now only displays names.

* Start typing contact name

Place Holder

Ana Holder

New

6. Add two more fields to this Type Ahead Block to store and view the data returned from the DataRaptor.

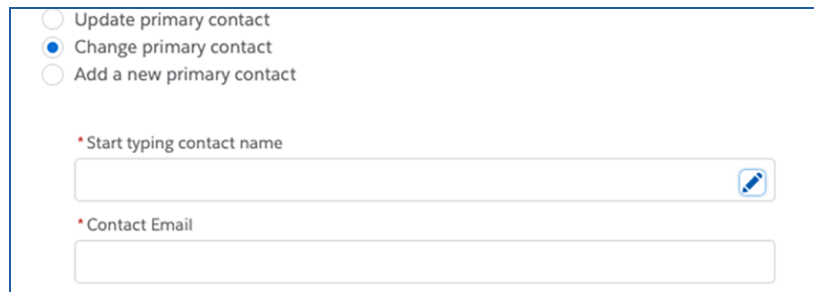
- In the header, click **Design**.
- In the **Build** panel, expand the **INPUTS** section.
- Drag an **Email** element into the TChangeContact type ahead block under the **DRChangeContactTA** element.
- Set the following values:

| Property | Value | Notes |
|-------------|--------------------|---|
| Name | ChangeContactEmail | This value matches the output field from the DataRaptor |
| Field Label | Contact Email | |
| Required | ✓ | |

- In the **Build** panel, drag a **Text** element under the **ChangeContactEmail** element, but still in the TChangeContact type ahead block.
- Change the **Name** to `ChangeContactId`. This specifies where to store the RecordId for the contact returned by the DataRaptor. Next, hide this field from the UI.
- Click **Edit Properties As JSON**.
- Change the hide value to `true`. Be sure to use all lowercase.
- Click **Close JSON Editor**.

7. Preview the use of the Type Ahead block.

- In the header, click **Preview**.
- Select **Change primary contact**.
- Click **Edit** (the pencil in the start typing field).



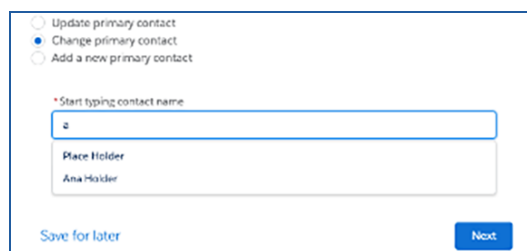
- d. Notice **Contact Email** is now displayed as a required value.
- e. Type **a** in the name field and select **Place Holder** from the list of choices.

Place Holder's name and email display.
- f. Review the Data JSON and notice the **ContactId** is present and available in the JSON data in the **BlkChangePriContact** node (You may have to expand the **TChangeContact-Block** to view the details), in the **ChangeContactId** field, which is used to save data when you make changes.

Task 2: Refine the UI of the Type Ahead Block

1. Notice two options for the Change primary contact fields:
 - a. The Edit button, which the user must click to display the Contact Email field.
 - b. The type-ahead field dropdown list, which shows a New option for adding a new contact, which you do not want to let end users do here.
2. Configure the OmniScript to refine the UI by hiding the Edit button and removing the New option.
 - a. In the header, click **Design**.
 - b. In the canvas, click the **TAChangeContact** type ahead element

If you're unsure of which element you're in, look at the properties title. It lists the element type.
 - c. In the **Properties** panel, for **New Item Label**, delete "New". This removes the **New** option from the typeahead dropdown list.
 - d. In the **Properties** panel, select **Edit Mode**, which specifies that all fields in the element are editable in the UI. This means that the user no longer needs to click **Edit** to view or update the contact email address.
 - e. Select **Hide Edit Button** to hide the button from the UI.
 - f. Click **Preview > Change primary contact**.
 - g. Notice the **Edit** button is removed.
 - h. Start typing in the field and notice the **New** option is removed.



Review

Confirm your understanding by answering these questions.

1. What are the three elements that need to be configured for a Type Ahead Block to work?
2. What three OmniScript elements are potential data sources for a Type Ahead Block?
3. Where do you configure what data is displayed in the Type Ahead dropdown?
4. What JSON node contains the text that the user types into a Type Ahead Block?
5. What are some ways to change the UI in OmniScript Elements?