

## Validate Data and Handle Errors



**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 my data entries to be validated and to receive warnings if I make mistakes.”*

Consider what kind of simple error checking helps them avoid making these mistakes. You'll start by simulating an error in your OmniScript, and work from there.

### Prerequisites

- Build an OmniScript with Branching
- Create a Type Ahead Block Element

### Tasks

1. Confirm Simple Error Checks
2. Add a Function to a Set Values Element for a Complex Error Check
3. Add Error Checking with a Set Errors Element

### Time

- 35 mins

---

## Task 1: Confirm Simple Error Checks

1. Simulate a simple error check in your OmniScript.
  - a. In Preview, select **Create a new primary contact** to view the associated fields.
  - b. Leave the fields empty and click **Next**.

The OmniScript displays an error under each field due to the missing field data. At least one visible field must be set to **Required** for these error messages to display. What do you do when fields are no longer visible, but required?

## Task 2: Add a Function to a Set Values Element for a Complex Error Check

1. Simulate a systematic error (the “Sad Path”).
  - a. In the header, click **Design**.
  - b. In the Canvas, click **IPGetAccountPriContactDetails**.
  - c. Update the **Active** toggle to closed. This deactivates the element, which is the data source.
  - d. Click **Preview**.
  - e. Notice the contact information fields are now empty for the default **Update primary contact** option.
  - f. Click **Next**. Notice the text under the Primary Contact Email field.
2. Add a condition that hides the Email field if the UpdateName field is blank.
  - a. In the header, click **Design** to return to the designer view.
  - b. In the Canvas, click the **Primary Contact Email** (UpdateEmail) element.

- c. Under **CONDITIONAL VIEW**, click the drop-down to review the **Condition Types**:

- i. Show Element if True
- ii. Disable Read Only if True
- iii. Set Element to Required if True

- d. In **View Condition**, click **Show Element if True**.

- e. Complete the fields as follows:

Enter the field name	Logical Condition	Enter the value
UpdateName	Does Not Equal	[leave blank]

- f. Click **Save**.
- g. Return to **Preview**.
- h. Notice the email field is no longer displayed. However, if the user clicks **Next**, they go to the end of the OmniScript without having completed any actions.
3. Add an element that adds the logic needed to stop people from moving forward without taking any actions.
- a. Return to the Design view.
  - b. In the **Build** panel, expand the **ACTIONS** section or use **Search** to locate **Set Values**.
  - c. Drag a **Set Values** action into the Canvas, under the **StepContacts** element. This may be easier if you collapse the **StepContacts** element first.
  - d. Set the following values for the Set Values action:
    - e. For **Name**, use `SVErrorCheck`. This is the variable name that shows up in the JSON node.
  - f. Under **Element Value Map**, click **+ Add New Element Value**.

- g. For **Element Name**, use `SVErrorCheck`.
- h. Under Value, type `=` to open Expression mode.
- i. Paste the following text into the Expression text box to form the logical statement:

```
IF (%UpdateName%=null&&%TChangeContact%=null&&%CreateLastNam  
e%=null, "NotOK", "OK")
```

- i. If this statement is true, the expression returns the string "NotOK".
- ii. If this statement is false, the expression returns the string "OK".
- iii. If any of the three names in the expression contain a value, this expression returns OK.
- iv. If all three names are missing a value, the expression returns NotOK.



**ALERT:**

If you paste the expression into the Value field rather than pasting it into the Expression text box (after you enter `=`), it does not work because `"="` doesn't prepend the formula. There must be an `"="` before the formula for it to be correctly evaluated.

Rather than copy the text from this document, copy the expression from your OmniScript in the **Navigate Action** Internal Notes field near the bottom of the properties panel.

- j. Click **Save**.
4. Test the element:
- a. Click **Preview > Next**.
  - b. In the **Data JSON** you see the value for "SVErrorCheck" is "NotOK".

```
21  ▶ StepContacts: Object
22    ▶ BlkUpdatePriContact: Object
23      UpdateEmail: null
24      BlkChangePriContact: null
25      BlkCreatePriContact: null
26      RadioPriContact: "Update"
27      SVErrorCheck: "NotOK"
```

- c. Click **Refresh**.
- d. Select **Change primary contact**, and type "h".
- e. Choose **Place Holder**, then click **Next**.
- f. Notice in the JSON code "SVErrorCheck" is "OK".

This means that the logic is working correctly.

```
21  ▶ StepContacts: Object
22    BlkUpdatePriContact: null
23    ▶ BlkChangePriContact: Object
24      ▶ TChangeContact-Block: Object
25      BlkCreatePriContact: null
26      RadioPriContact: "Change"
27      SVErrorCheck: "OK"
```

## Task 3: Add Error Checking with a Set Errors Element

1. Add a Set Errors element to send users back to a previous step if data is incomplete.
  - a. In the header, click **Design**.
  - b. In the **Build** panel, expand the **ACTIONS** section or use **Search** to locate **Set Errors**.
  - c. Drag the **Set Errors** element into the canvas, under the SVErrorCheck element.
  - d. Change Name to `SetErrors`.
  - e. Under **Element Error Map**, click + **Add New Element Value**.
  - f. For **Element Name**, select **RadioPriContact**.
  - g. For Value, add `Please change or create a new primary contact!`

You need to specify the action to take when the radio button element is displayed. In this case, you want the user to either change or create a new primary contact.

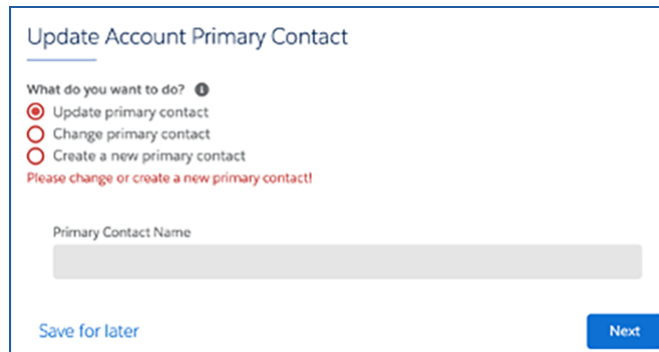
- h. Click **Save**.
2. Add a conditional view to the Set Errors element.

Set Errors elements must have a conditional view. The condition for this error check is that SVErrorCheck is equal to NotOK. Add the condition as follows:

- a. Under **CONDITIONAL VIEW > View Condition**, click **Show Element if True**.
- b. For the field name, choose **SVErrorCheck** from the dropdown list.
- c. Ensure that **Equal To** is selected.
- d. For the **Value**, use `NotOK`. (OK is case sensitive.)
- e. Click **Save**.

Note the condition icon (the orange eye) now displays.

3. Test the Set Errors element.
  - a. In the header, click **Preview**.
  - b. Confirm no primary contact is shown and click **Next**.
  - c. Notice the instructions in red.



- d. Select **Change primary contact**.
  - e. Type **h** and choose **Place Holder**.
  - f. Click **Next**.
  - g. In the **Data JSON** notice that the value for “SVErrorCheck” is “OK”.

```
21  > StepContacts: Object
22    BlkUpdatePriContact: null
23    > BlkChangePriContact: Object
24      TChangeContact-Block: Object
25    BlkCreatePriContact: null
26    RadioPriContact: "Change"
27    SVErrorCheck: "OK"
```

4. Activate the Integration Procedure element again.
  - a. Return to the Design view.
  - b. In the Navigation Panel, Tree View, select the **IPGetAccountPriContactDetails** element to show the properties.
  - c. Change the **Active** toggle to blue or “on”.

- d. Click **Preview**. You see the default selection (Update Primary Contact) with the Primary Contact data displayed again.

## Review

Confirm your understanding by answering these questions.

1. What is the requirement for a step to have simple error checking?
2. What options are there for creating complex logical conditions in an OmniScript?  
What are advantages and disadvantages for each?
3. What settings are required for a Set Errors Element?