Incident Resolved/ Closed UI Policy

Lab 3.1

20-25 minutes

Lab objectives

You will perform the following in this lab:

- Write, test, and debug a UI Policy.

  The UI policy should:
  - Trigger when the State is set to Resolved.
  - Change Urgency and Impact fields to read-only.
  - Change the Resolved by field to mandatory.
  - Display an Info Message.
- Complete a Cloud Dimensions' Phase II requirement.

Business Problem

When closing Incident records, IT Analysts are changing the value of the Impact and Urgency fields to manipulate SLA results.

In addition to the baseline mandatory fields when a record is put in a Resolved or Closed state, the Incident Management team would also like to consistently capture who is resolving Incidents for reporting purposes.

IT Analysts do not always remember to populate the Closure Information fields as they are on a different tab and are frustrated with the number of times they are presented with the 'Mandatory fields were missed' alert after they attempt to save the record. They have requested a message on the form, reminding them which fields are mandatory before they attempt to save the record.

Project Requirement

Because every requirement is based on the same Incident condition, only one UI Policy is required to solve all the business problems. When an Incident's state is Resolved or Closed:

- Make the Impact and Urgency fields read-only.
- Make the Closed by field mandatory.
- Configure an Info Message reminding users which fields are mandatory before they attempt to save a record.

A. Create a UI Policy

1. Navigate to **System UI > UI Policies.**

2. Create a new UI Policy.

   Table: **Incident [incident]**
   Active: **Selected (checked)**
   Short Description: **Lab 3.1 Incident Resolved or Closed**
   Order: **100** (if not available, configure the form to include the **Order** field)
   Condition: **State | is one of | Resolved + Closed**
   (Hold the 'Shift' key down to select both items in the list)
   Global: **Selected (checked)**
   On load: **Not selected (not checked)**
   Reverse if false: **Selected (checked)**
   Inherit: **Not selected (not checked)**

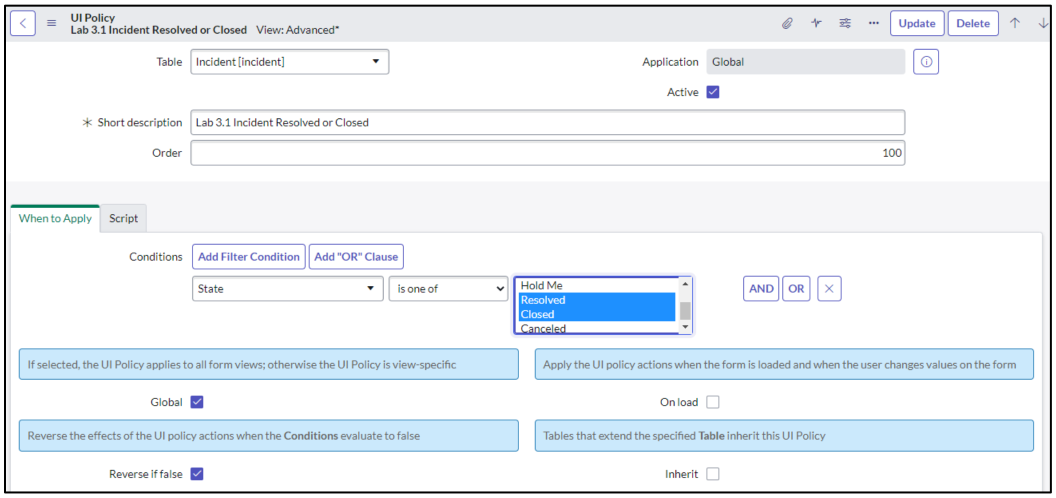

UI Policy
Lab 3.1 Incident Resolved or Closed View: Advanced*

Table: Incident [incident]
Application: Global
Active: ✓
Short description: Lab 3.1 Incident Resolved or Closed
Order: 100

Conditions

State | is one of | Hold Me, Resolved, Closed, Canceled

If selected, the UI Policy applies to all form views; otherwise the UI Policy is view-specific
Global: ✓

Apply the UI policy actions when the form is loaded and when the user changes values on the form
On load: ☐

Reverse the effects of the UI policy actions when the **Conditions** evaluate to false
Reverse if false: ✓

Tables that extend the specified **Table** inherit this UI Policy
Inherit: ☐

3. **Save** the record to remain on the form.

4. Add a UI Policy Action.

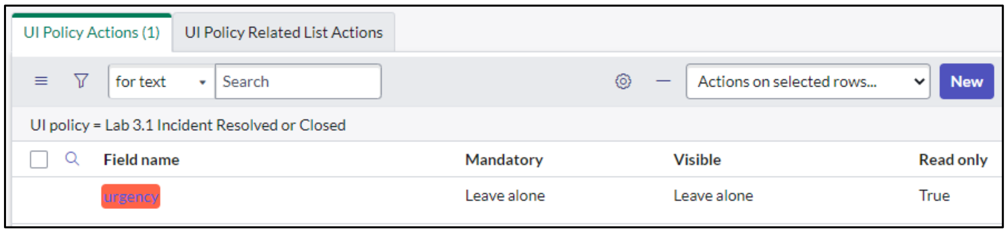
   a) Scroll down to the UI Policy Actions Related List and select the **New** button.

   b) Configure the UI Policy Action:

      Field name: **Urgency**
      Mandatory: **Leave alone**
      Visible: **Leave alone**
      Read Only: **True**

   c) Select **Submit**.

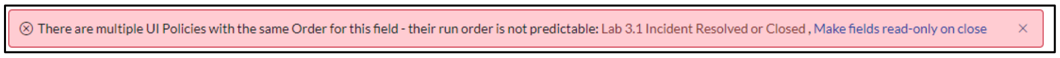
   d) Examine the record you just created in the list of *UI Policy Actions*. Notice that **urgency** is highlighted in red? This indicates an error that requires attention. Select the **urgency** link to open the record.



UI policy = Lab 3.1 Incident Resolved or Closed

Field name	Mandatory	Visible	Read only
urgency	Leave alone	Leave alone	True

   e) An error message appears indicating there are multiple UI Policies with the same Order for this field. Select the **Make fields read-only on close** link in the error message to open the conflicting UI Policy record.



There are multiple UI Policies with the same Order for this field - their run order is not predictable: Lab 3.1 Incident Resolved or Closed , Make fields read-only on close

   f) If the *Default* view of the form opens, select the **Advanced view** Related Link to display the Order field. Record the **Order** of the *Make fields read-only on close* UI Policy here:

      ____

   g) Return to the **Lab 3.1 Incident State Resolved or Closed** UI Policy. If the *Default* view of the form opens, select the **Advanced view** Related Link.

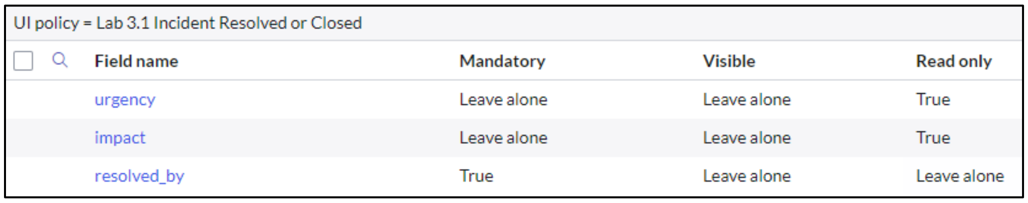
   h) Change the **Order** field value to a value greater than the conflicting UI Policy Order field's value.

   i) **Save** the record to remain on the form.

   j) Notice that **urgency** is no longer highlighted in red.

5. Add two additional UI Policy Actions:

   Resolved by: **Mandatory**
   Impact: **Read only**



UI policy = Lab 3.1 Incident Resolved or Closed

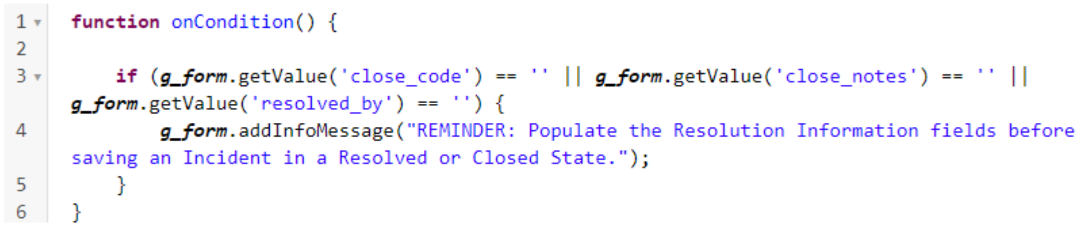
Field name	Mandatory	Visible	Read only
urgency	Leave alone	Leave alone	True
impact	Leave alone	Leave alone	True
resolved_by	True	Leave alone	Leave alone

6. On the *Script* tab, select (check) the **Run scripts** field.

7. Examine the following pseudo-code for the **Execute if true** script you will write:

```
• When the condition is true
    o If the Closed code, Closed notes or Resolved by fields do not
      have values
        ▪ Display an Information Message reminding the Analysts of
          mandatory fields required before saving the record.
```

8. Write the **Execute if true** script:



```
function onCondition() {

    if (g_form.getValue('close_code') == '' || g_form.getValue('close_notes') == '' ||
g_form.getValue('resolved_by') == '') {
        g_form.addInfoMessage("REMINDER: Populate the Resolution Information fields before
saving an Incident in a Resolved or Closed State.");
    }
}
```

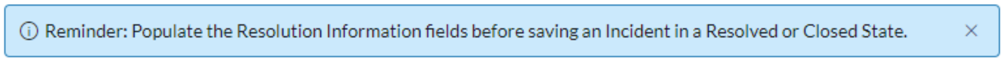
9. Select **Update**.

B. Test Your Work

1. Open an existing Incident.

2. Set the value of *State* to **Resolved** or **Closed**.

3. Did the Information Message appear at the top of the form? If not, debug and re-test.

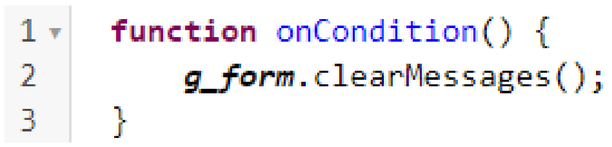


Reminder: Populate the Resolution Information fields before saving an Incident in a Resolved or Closed State.

4. If the user updating a record selects one of these two states in error and immediately sets the value of *State* back to **In Progress**, does the Info Message at the top of the form disappear?

5. Open the **Lab 3.1 Incident Resolved or Closed** *UI Policy*.

6. Write the **Execute if false** script:



```
function onCondition() {
    g_form.clearMessages();
}
```

7. Select **Update**.

8. Open an existing Incident.

9. Set the value of the State field to **Resolved** or **Closed**.

10. Once the Info Message appears at the top of the form, update the value of *State* to **In Progress**. Does the Info Message at the top of the form disappear? If not, debug and re-test.

11. Does the **Reverse if false** field need to be selected for the **Execute if false** script to execute?

    ____

12. When the **Reverse if false** field is selected and the **Execute if false** field is empty, does the reverse of what is scripted in the Execute if true field occur?

    ____

    ____

Lab Completion

Great job! You successfully created a UI Policy that works everywhere in the platform.