Lab Guide

Advanced Communications Using SMS and Voice

Michael Ritchie & Michael Slabodnick

Default Login / Password:

admin / Knowledge17

itil / Knowledge17

employee / Knowledge17

Lab Sponsored by Microsoft Azure



This
Page
Intentionally
Left
Blank

Lab Goal

Lab 1.0
Verify Notify Properties

This lab explains how to verify that your instance is communicating properly with Twilio. This is a prerequisite for all other labs in this guide.

Twilio API credentials were automatically provisioned to your instance.

Log in to the Training Instance

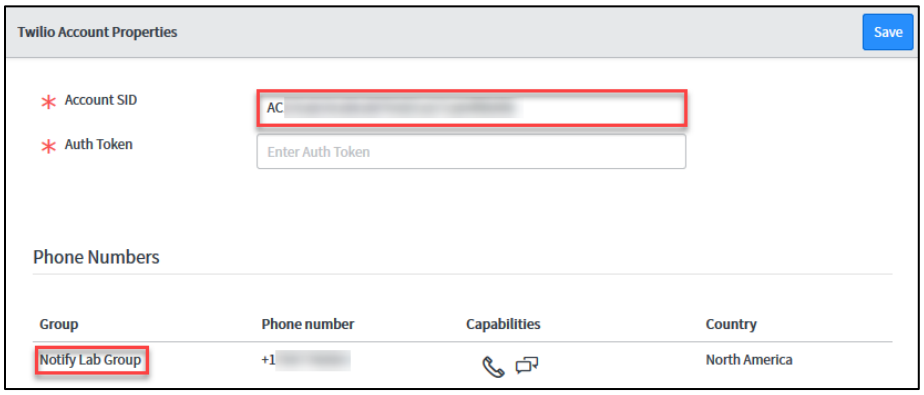
1. Navigate to the unique instance URL provided.
2. Log in with the provided administrator credentials.

View the Notify Application Properties Page

1. In the Filter navigator box, type **notify**.
2. Click **Twilio Configuration** in the Notify Application under Administration.
3. View the configuration details.

Lab Success Verification

1. Verify that the **Account SID** is populated and that you have a Phone Number listed at the bottom assigned to the Notify Lab Group.



2. If for some reason these fields are not populated or you do not have an assigned phone number, please raise your hand for assistance.

Lab Goal

Lab 1.1 Update Users

This lab explains how to update an out-of-the-box user with your mobile phone for use in this lab.

Update User Record Mobile Phone Field Type and Enter Mobile Phone

For ease of use and to demonstrate the capabilities of ServiceNow Notify, a workflow has been set up in your instance to automatically update a few demonstration user records with your mobile phone by texting the word “phone” to your instance’s Notify telephone number. Joe Employee and Beth Anglin’s user records are updated with your mobile phone number.

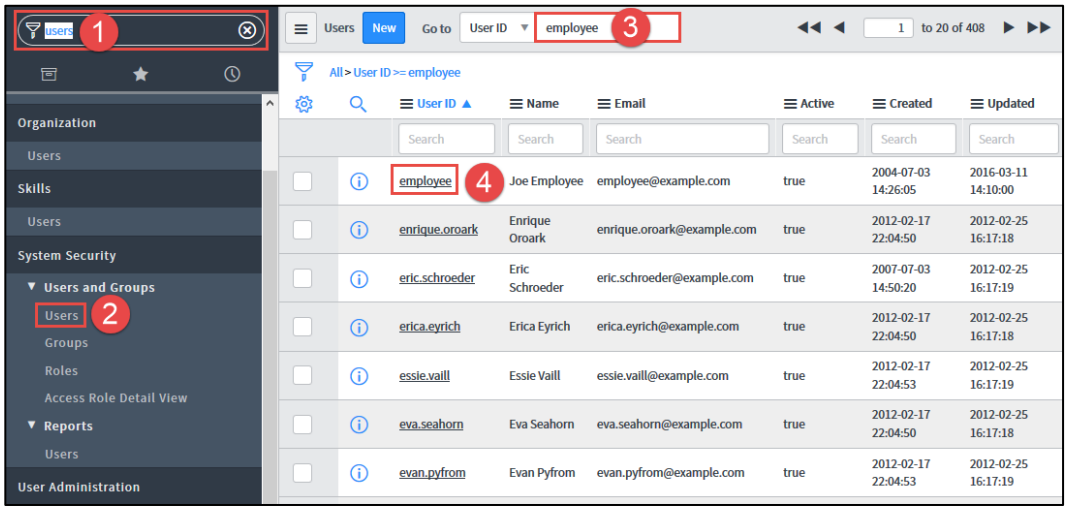
1. In the Filter navigator box, type **notify**.
2. Click **Numbers** under Notify Application.
3. Only one number has been provisioned to your instance, click the phone number to view the record.
4. This record contains the specific Twilio number assigned to your instance. Using your mobile phone, send the word “**phone**” via SMS to your assigned phone number.



5. Momentarily you should receive a SMS stating “All users in your instance have been successfully updated.” If you receive this SMS message you may skip the following section and proceed to reporting Lab Progress.

If you did not receive the success message above and but instead received the following message, please continue with the steps below: “There was a problem updating the users in your instance. Please follow steps to update manually.”

6. In the Filter navigator box, type **users**.



7. Select **Users** under the System Security application.

8. Using the list search at the top-middle of the screen, search for the Name **Joe**.

9. Click **employee** in the list.

10. Notify requires that phone numbers be in E.164 format. Notice that the mobile Phone field has been updated to a field type of “Phone Number (E.164)” and that you have a region field along with the phone number. Choose your personal region and **enter your personal mobile number** in the field below it. Format does not matter; ServiceNow automatically corrects it to the proper format with next step.

    Note: Out-of-the-box, only a few regions appear. For US and Canadian phones, choose North America. If you do not find your region in the drop-down list, please raise your hand for help enabling your region.

11. After you click out of the **mobile phone** field, ServiceNow converts it to the proper format.

12. Click **Update** to save Joe’s user record.

13. Update one more user for later use in the Lab. Use the list search at the top middle of your screen to search for the name **Beth Anglin**.

14. Click **beth.anglin** in the list.

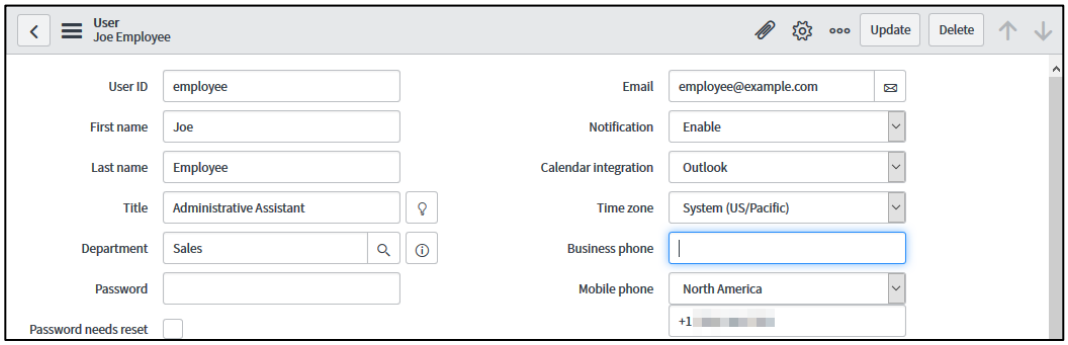
15. In the Mobile phone field, choose your personal region and enter your mobile number in the field beside it. After you click out of the mobile number field, ServiceNow converts it to the proper format.

    Note: For US and Canadian phone numbers, choose North America.

16. Click **Update** to save Beth's user record.

Lab Success Verification

1. Joe Employee's and Beth Anglin's user records have been updated with your mobile phone.



Lab Goal

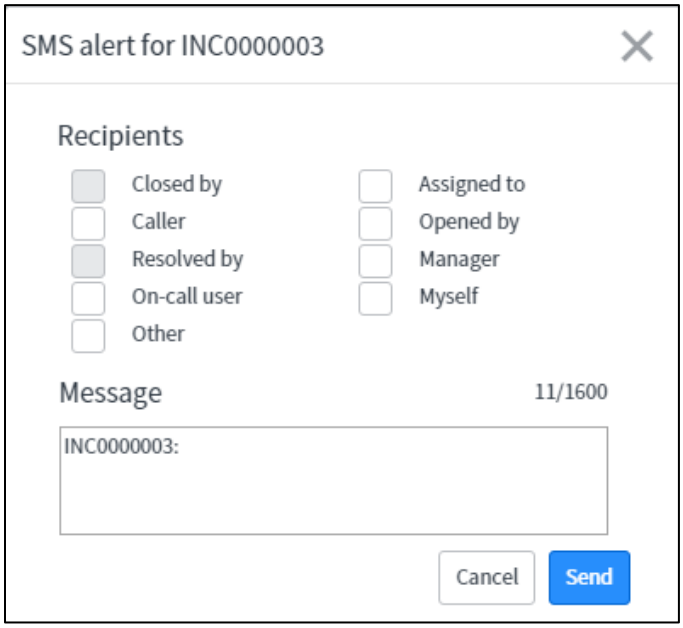
Lab 2.0 Send SMS from Incident

This lab explains how to use Notify to send an SMS on any task-based table.

ServiceNow provides the ability to send SMS and create conference calls on any task-based table.

Send an SMS from an Existing Incident

1. In the Filter navigator box, type **incident**.
2. Click **Open** under the Incident application.
3. In the list on the right, search for and click **INC0000003** that is assigned to the Network team and Joe Employee is the Caller.
4. Scroll toward the bottom to the Related Links above the Task SLAs list. Click **Send SMS**.
5. A dialog appears similar to the following:



6. Notice that all reference fields to the User (sys_user) appear in this dialog. This dialog also includes any user reference fields you create on your own through configuration. Any field that is not populated has a grayed-out checkbox meaning that you cannot send an SMS to those individuals.

7. Click the checkbox beside **Caller** and notice the Caller text automatically changes to the caller of the Incident, Joe Employee in this example.

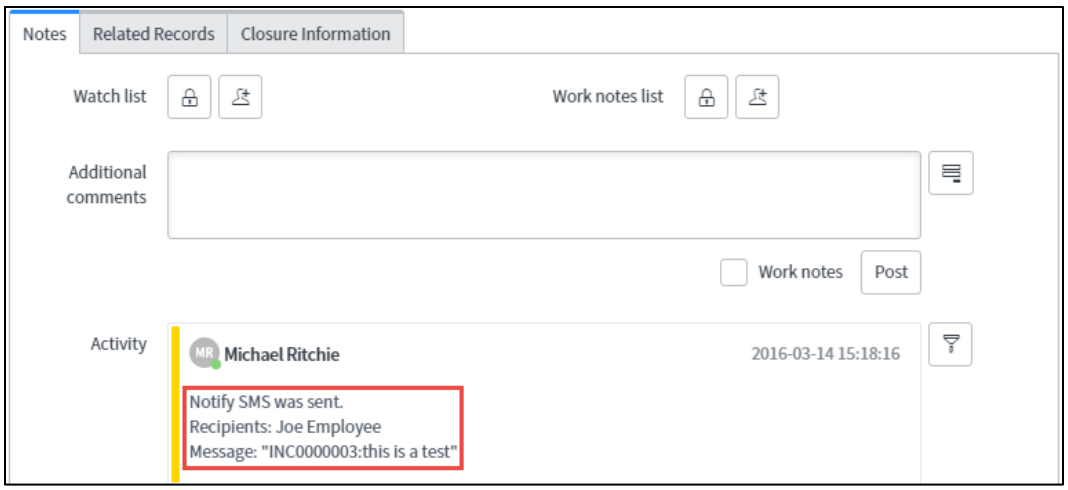
8. Click the checkbox beside **On-call user** and notice that a name is automatically populated. This option shows up automatically if the On-Call Scheduling plugin is enabled. This dialog automatically looks up the primary person who is on-call for the task's assignment group and allows you to send them an SMS.

9. Notice there is also an **Other** option which allows you to include any user in your ServiceNow user table. If selected, it also allows you to enter any E.164 number for phones that are not in your user table (e.g., vendors, home phones). Numbers must be entered in E.164 format to be accepted (i.e., +18005551212).

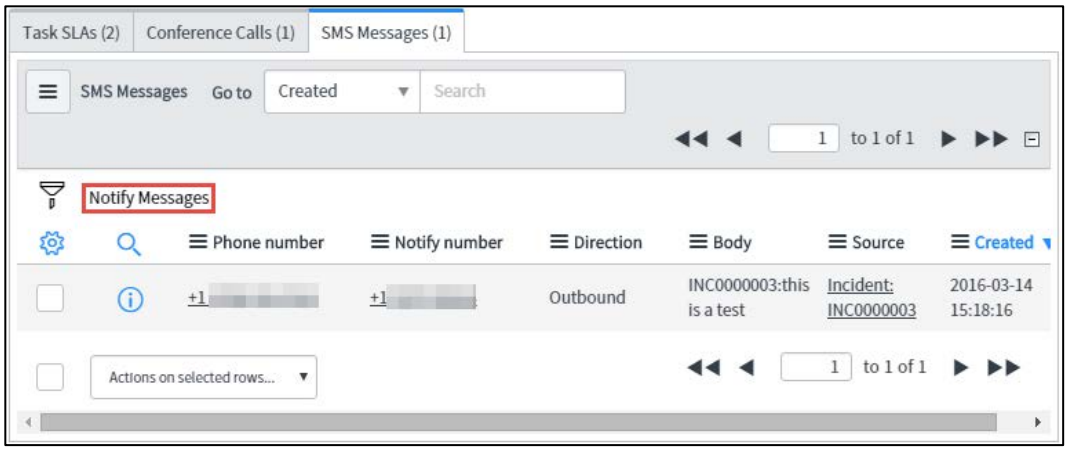
10. By default, the Message begins with the task number and from there you can enter additional details up to 1,600 characters. Please enter an additional message.

11. Since most of us only have one phone, **only check the Caller** (Joe Employee) since we previously set his mobile phone with our mobile phone number. Click **Send** at the bottom.

12. An SMS is sent momentarily to your mobile phone. The SMS is logged directly as an Incident activity in the Notes Section.



13. View all SMS messages sent from the Incident by scrolling to the bottom of the form and selecting the SMS Messages tab.
    Note: you may need to click **Notify Messages** to refresh the list for the SMS message to appear.



Lab Success Verification

1. An SMS was received from an Incident.

Lab Goal

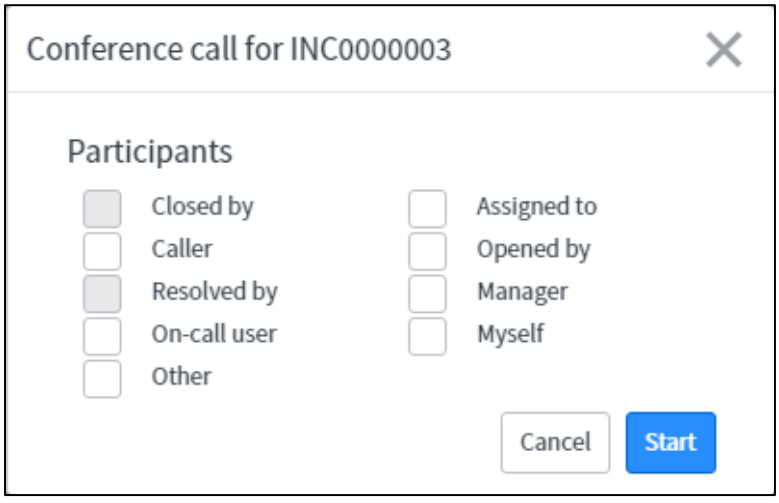
Lab 3.0 Start and Manage Conference Call

This lab explains how to use Notify to start a Conference Call on any task-based table.

ServiceNow provides the ability to send SMS and create conference calls on any task-based table.

Initiate a Conference Call from an Existing Incident

1. In the Filter navigator box, type **incident**.
2. Click **Open** under the Incident application.
3. In the list on the right, click **INC0000003** that is assigned to the Network team and Joe Employee is the Caller.
4. Scroll toward the bottom to the Related Links above the Task SLAs list. Click **Start Conference Call**.
5. A dialog appears similar to the example.



6. Notice that all reference fields to the User (sys_user) appear in this dialog. This dialog also includes any user reference fields you create on your own through configuration. Any field that is not populated has a grayed-out checkbox, meaning that you cannot invite them to a conference call.

7. Click the checkbox beside **Caller** and notice the Caller text automatically changes to the caller of the Incident, Joe Employee in this example.

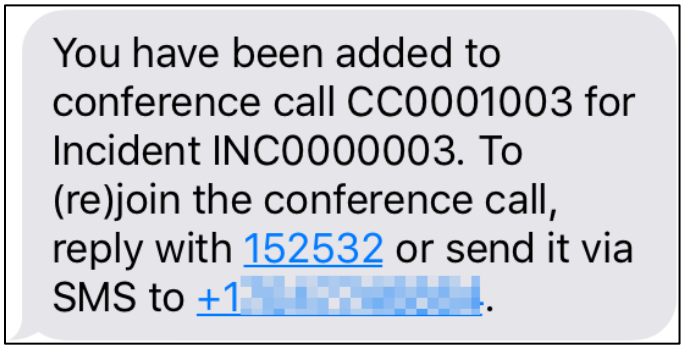
8. Click the checkbox beside **On-call user** and notice that a name is automatically populated. This option shows up automatically if the On-Call Scheduling plugin is enabled. This dialog automatically looks up the primary person who is on-call for the task's assignment group and allows you to invite them to the conference call.

9. Notice there is an **Other** option that allows you to invite any user in your ServiceNow user table. It also allows you to enter any E.164 number for phones that are not in your user table (e.g., vendors, home phones). Numbers must be entered in E.164 format to be accepted (i.e., +18005551212).

10. Since most of us only have one phone, **only check the Caller** (Joe Employee) since we previously set his mobile phone with our mobile phone number. Click **Start** at the bottom.

11. Your mobile phone receives a call momentarily. Notice that Twilio has text-to-speech capabilities and can read you some details about the Incident. This is all controlled by a workflow that can easily be configured to add additional information. When prompted, press **1** to accept the call. **Please remain on the call for the next few steps**.
Note: Twilio plays hold music until more than one participant joins the call.

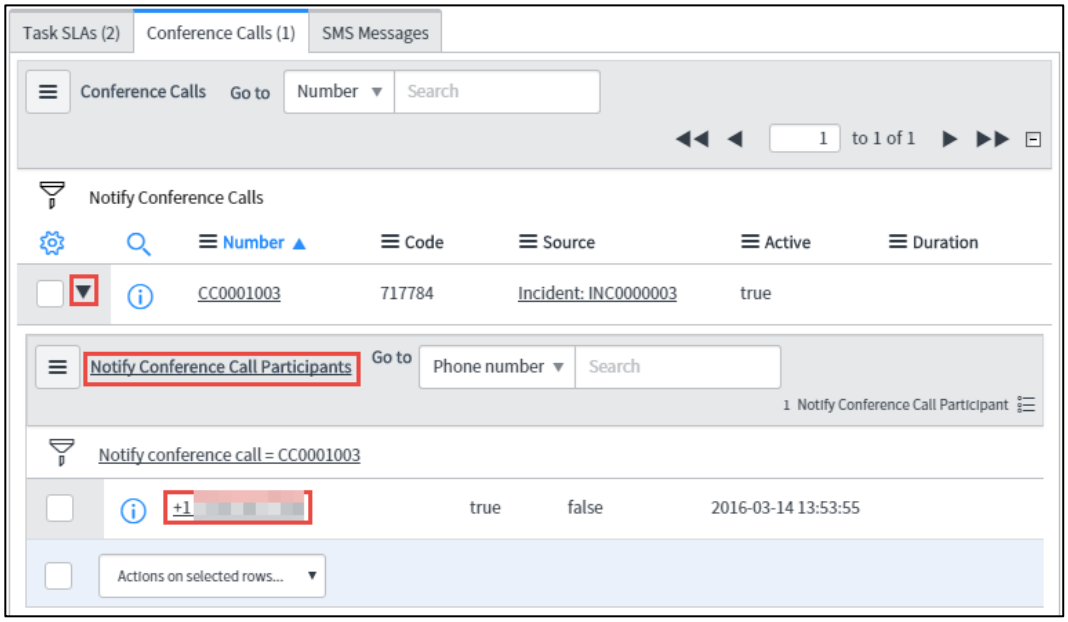
12. You also receive an SMS message on your phone, confirming your invitation to the conference call with details on how to join back or invite other people.



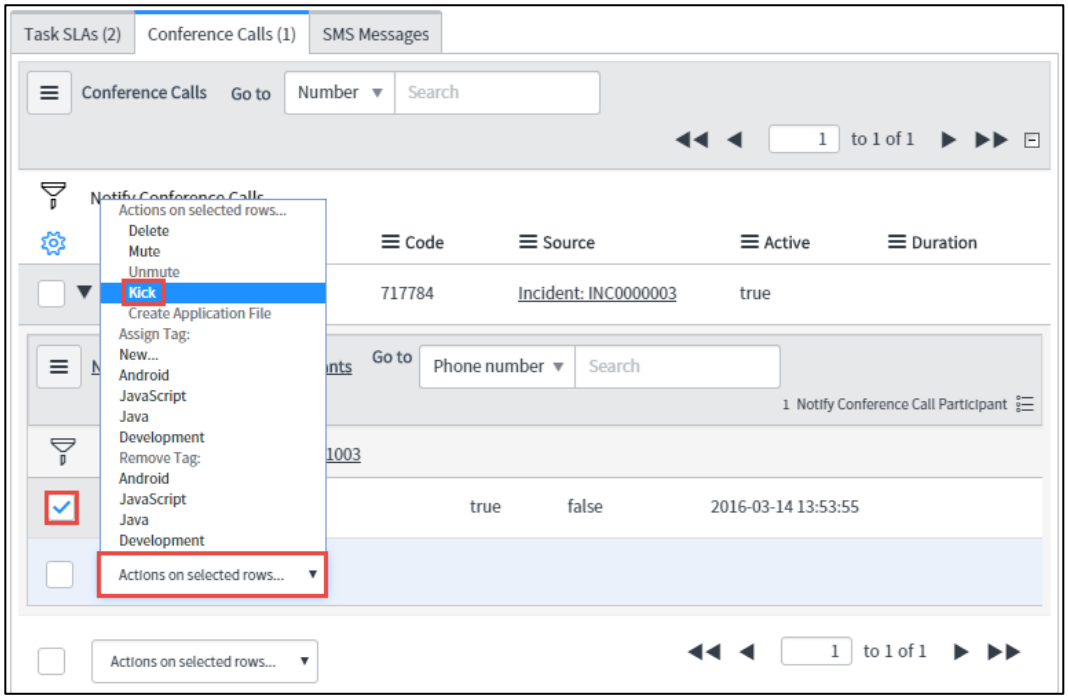
Note: Users can also call the Twilio phone number provided and enter the code to join the call.

13. Once a conference call has started, you can **mute**, **unmute**, and **kick** participants from the call directly from the task record that initiated it. This process is covered next.

14. **You are still on the Conference Call**. Scroll to the bottom of the Incident form to the Conference Calls section (click **Conference Calls** tab). Notice a conference call record is created to reflect that a conference call was initiated from this incident. The code is also available in case others want to join. Expand the hierarchal list to view the participants on the conference bridge.

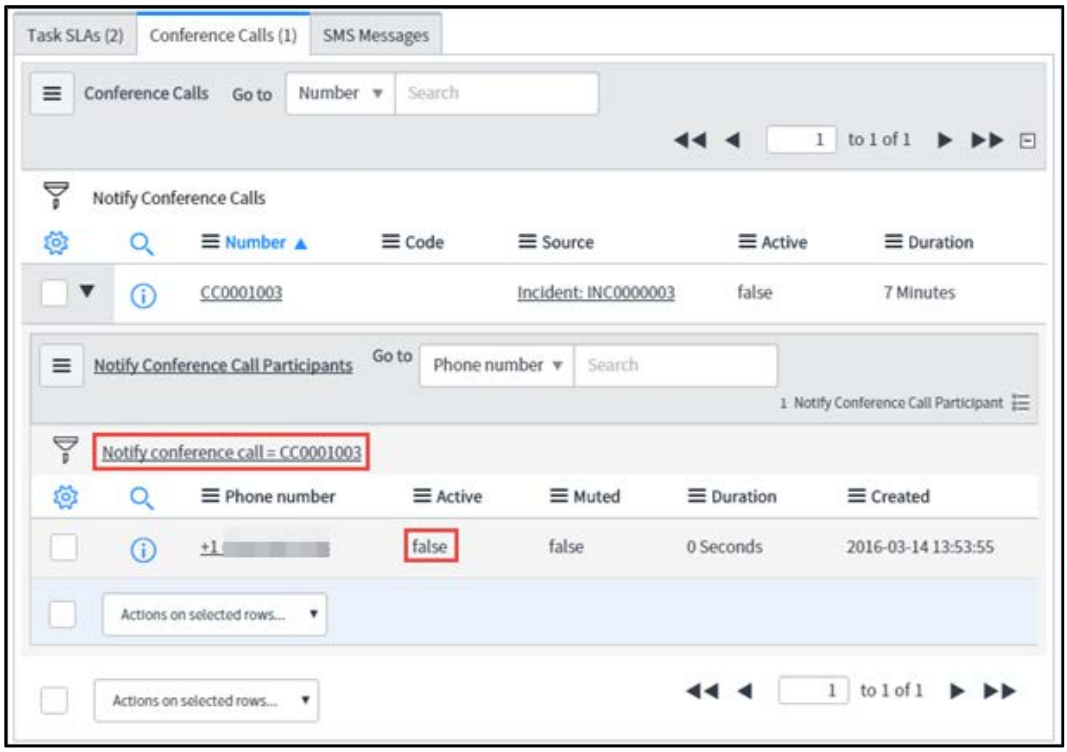


15. This list contains options to **mute**, **unmute** or **kick** the participant from the call. Click the **checkbox** on the row with your number and click the **Actions on selected rows** drop-down list. Choose **Kick**.



16. You should hear a message that you are being disconnected, and the call should end.

17. Refresh the **Participants** list by clicking the **Notify conference call = CC000100X** link. You should see that you are no longer active in the conference call.



Lab Success Verification

1. A conference call was successfully initiated from an Incident.
2. You removed a participant from the conference call.

Lab Goal

Lab 4.0 Initiate Call from Task

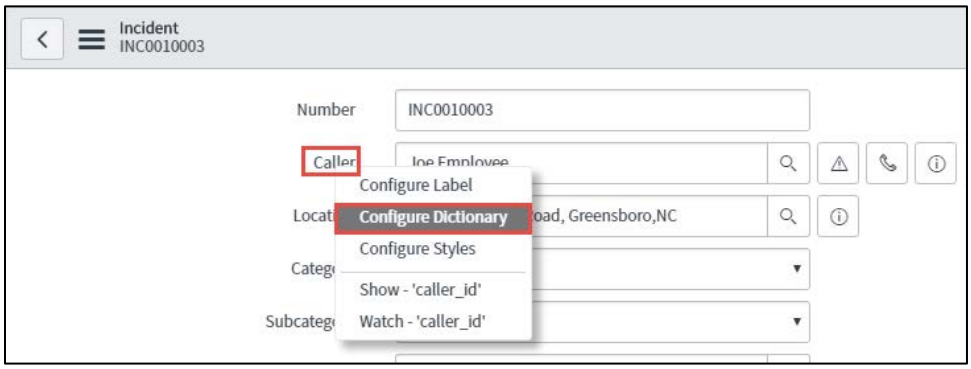
This lab explains how to enable a phone button beside the Incident Caller field that allows the user to connect to the Caller directly from your instance.

ServiceNow supports OpenFrame and WebRTC. These technologies permit IVR capabilities directly in ServiceNow. By enabling the Openframe (com.sn_openframe) and CTI Softphone (com.snc.cti) plugins in your instance along with Notify, agents can make calls directly from the instance.

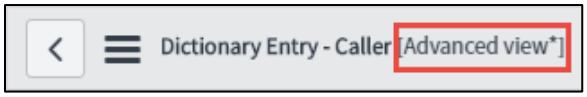
Note: WebRTC is only supported in **Chrome** and **Firefox**. If you do not have one of these browsers, you can skip this lab.

Add UI Macro to Caller and Initiate Call

1. In the Filter navigator box, type **incident**.
2. Click **Open** under the Incident application.
3. In the list on the right, click **INC0000003** that is assigned to the Network team and **Joe Employee** is the Caller.
4. Right-click the **Caller** field label and choose **Configure Dictionary**.



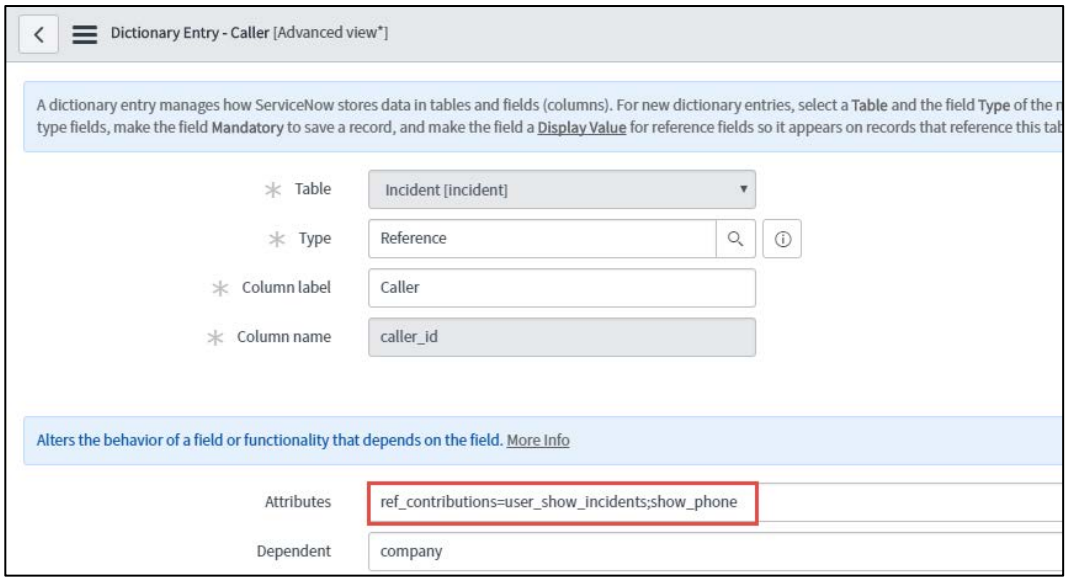
5. When the **Dictionary Entry – Caller** form appears, make sure you see the Advanced view. If not, scroll to Related Links and click **Advanced view**.



6. In the Attributes field, you should see **ref_contributions=user_show_incidents**. This is a UI Macro that shows an additional button beside the **Caller** field to show you the caller's incident. Add a new button to make a call from the instance.

7. Multiple UI Macros can be added by separating them with a semicolon. The CTI Softphone includes a UI Macro, **show_phone**, that you can leverage. Enter the following into the Attributes field:

   ref_contributions=user_show_incidents;show_phone



8. Click **Update**.

9. You should now return to INC0010003; if not, navigate to it again following **Steps 1-3**.

10. A new phone button should appear beside the **Caller** field.



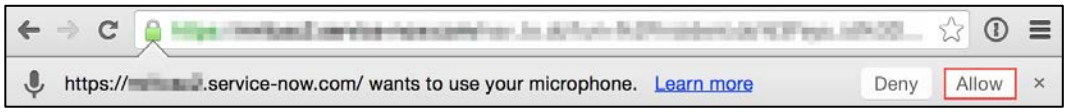
11. Ensure that your volume and microphone on your computer are active. Click the **Call** button.

    Note: You need to share your microphone with your browser before the call can take place.

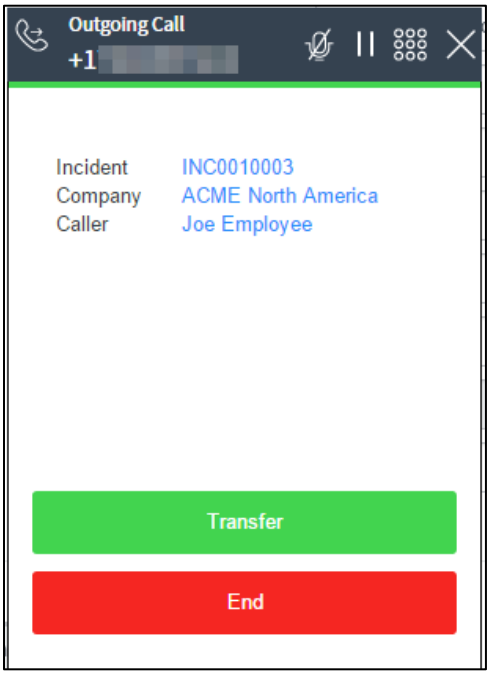
    Firefox presents a pop-up dialog like the example.



    Chrome presents a pop-up like the example.



12. Shortly the phone should ring and the following dialog appears, allowing you to end the call.



13. After the call ends, an update also appears in the Incident, confirming the call and the duration of the call.



14. The show_phone UI macro can be added to any User (sys_user) reference field.

Lab Success Verification

1. You were able to call your phone from an Incident record.

Lab Goal

Lab 5.0 Assignment Group Conference Call

This lab explains how to use the Notify API to enhance incident management by quickly initiating a conference call from an assignment group field on the incident.

The Notify API is available for developers to enhance calling functionality across the entire ServiceNow platform. With the use of AJAX client script functionality, this lab demonstrates an example of calling Notify from a form using a UI Macro.

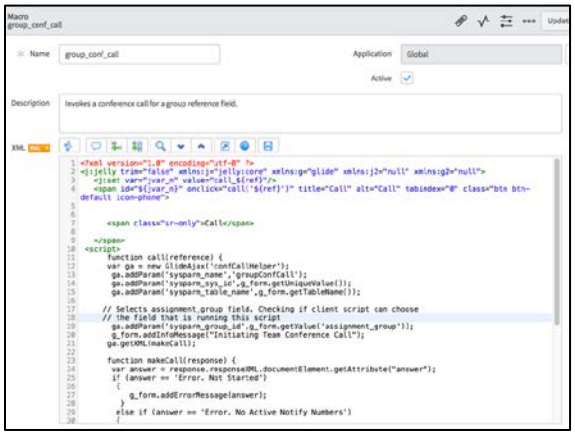
Note: The UI macro and script include for this lab have already been created for your lab instance. For installation onto a non-lab instance, the update set is available on ServiceNow share at https://share.servicenow.com.

Validate the UI Macro and Script Include

1. In the Filter navigator box, type **ui macros**.
2. Click **UI Macros** in the **System UI** Application.
3. In the search box above the name column, type **group_conf_call**.



4. Click on **group_conf_call** from the list to navigate into the record.



5. In the UI Macro, under the <script> code, look for a line calling the Script Include **confCallHelper**.

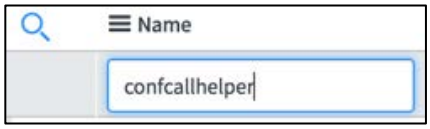
```
function call(reference) {
var ga = new GlideAjax('confCallHelper');
 ga.addParam('sysparm_name','groupConfCall');
 ga.addParam('sysparm_sys_id',g_form.getUniqueValue());
 ga.addParam('sysparm_table_name',g_form.getTableName());
```

Note: The UI Macro, as client code, calls the Script Include in GlideAjax. The function called in the Script Include is **groupConfCall**. Its value is set as a system parameter.

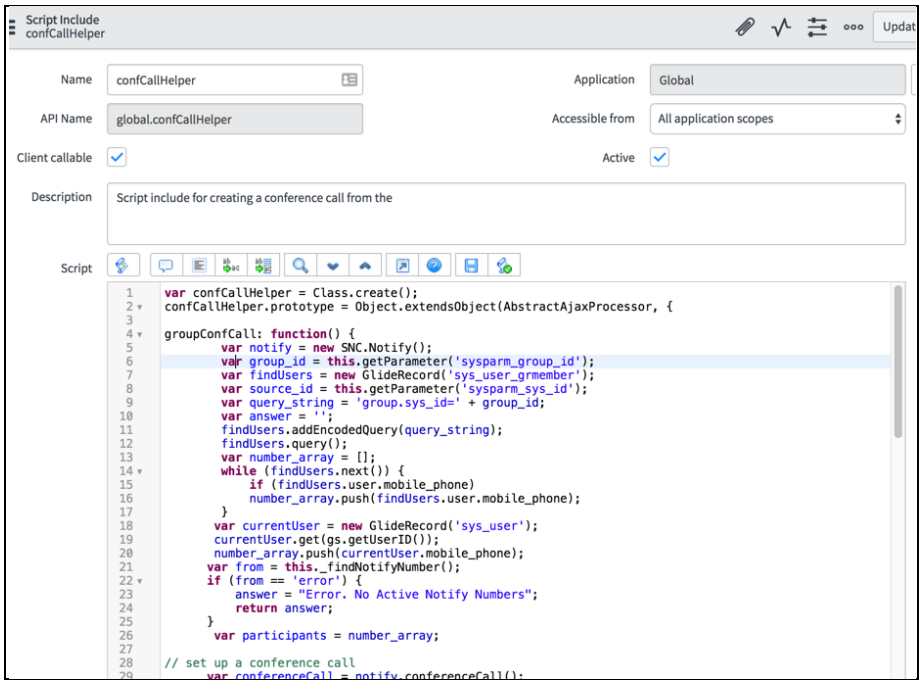
6. In the navigator box, type **script includes**.

7. Click **Script Includes** in the **System Definition** Application.

8. In the search box above the name column, type **confcallhelper**.



9. Click on **confCallHelper** from the list to navigate into the record.



10. In the confCallHelper script include, look for the code declaring a new conference call.

```
// set up a conference call
      var notify = new SNC.Notify();
      var conferenceCall = notify.conferenceCall();
```

Note: The API documentation on working with conference calls in Notify can be found at https://developer.servicenow.com/app.do#!/api_doc?v=istanbul&id=r_N-call_S_S.

Configure the Group Conference Call on Incident Assignment Group

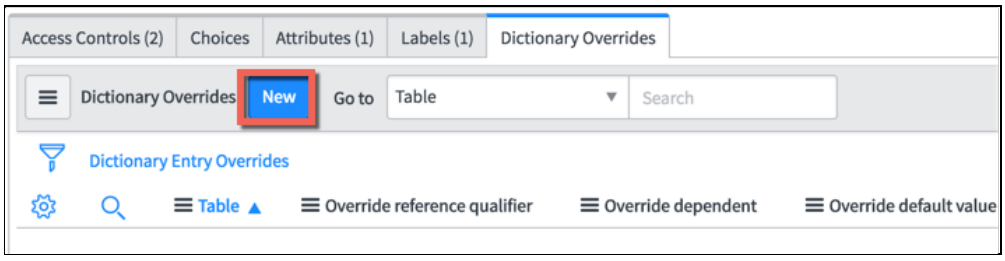
1. In the navigator menu, click **Incident -> Open**.
2. Click **New** to create a new incident.



3. Right-click the **Assignment Group** field label and left-click **Configure Dictionary**.



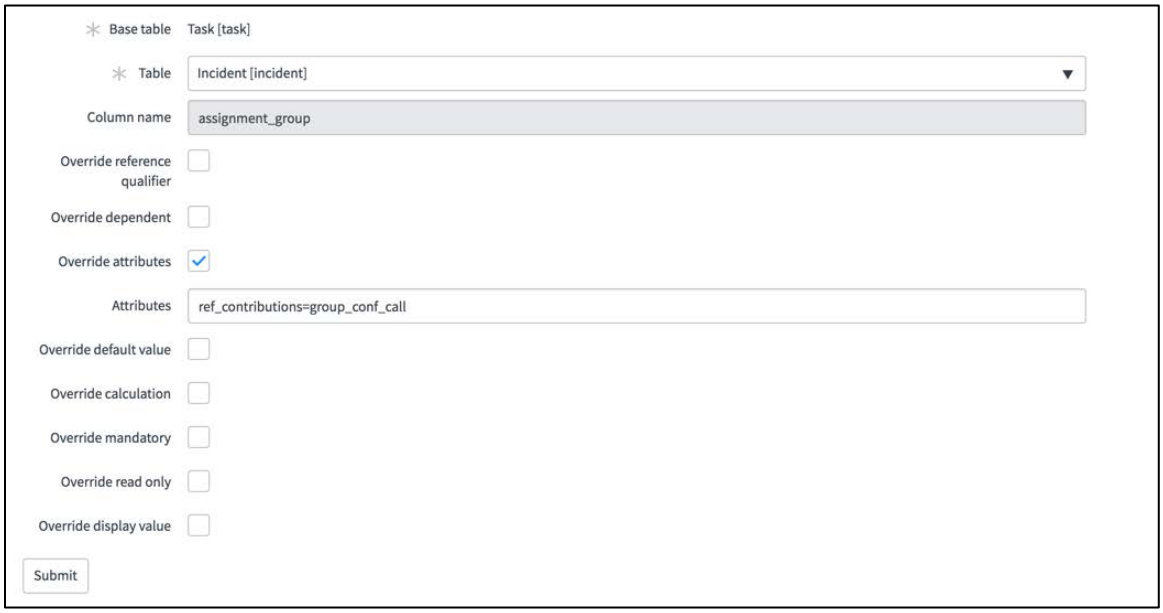
4. In the dictionary for the assignment_group field, scroll to the related list section and find **Dictionary Overrides**.
5. Click **New** to create a new Dictionary Override.



Note: The dictionary override allows a group conference call button to be shown **only** for incidents. Configuring the UI Macro on the task table displays the button on all tables.

6. Enter the following information:

   Table: **Incident**
   Dictionary attributes: **True**
   Attributes: **ref_contributions=group_conf_call**.



7. Click **Submit** to save the attribute and be taken back to the dictionary record.

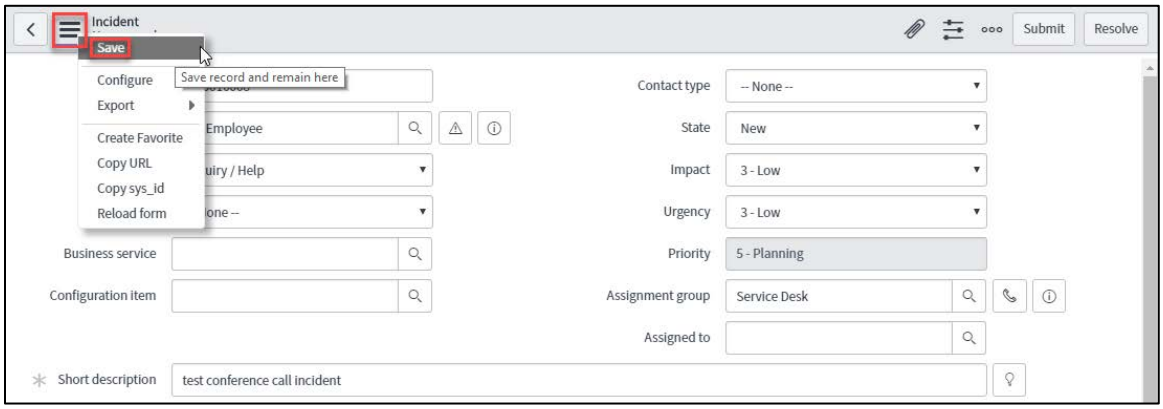
8. Click **Update** on the dictionary record to return to the new incident form.

9. Enter the following information:

   Caller: **Joe Employee**
   Short Description: **test conference call incident**
   Assignment Group: **Service Desk**



10. Click **Additional actions** and select **Save** to remain on this record.

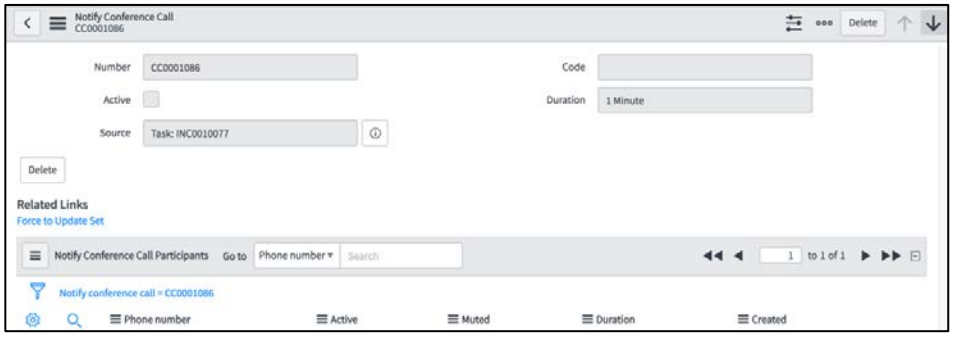


11. Next to Assignment Group, click the **phone icon** to initiate a conference call with that group.

Lab Success Verification

1. Verify that you receive a phone call and SMS text message asking you to join a conference call.

2. Verify that you're redirected to a conference call record.



Lab Goal

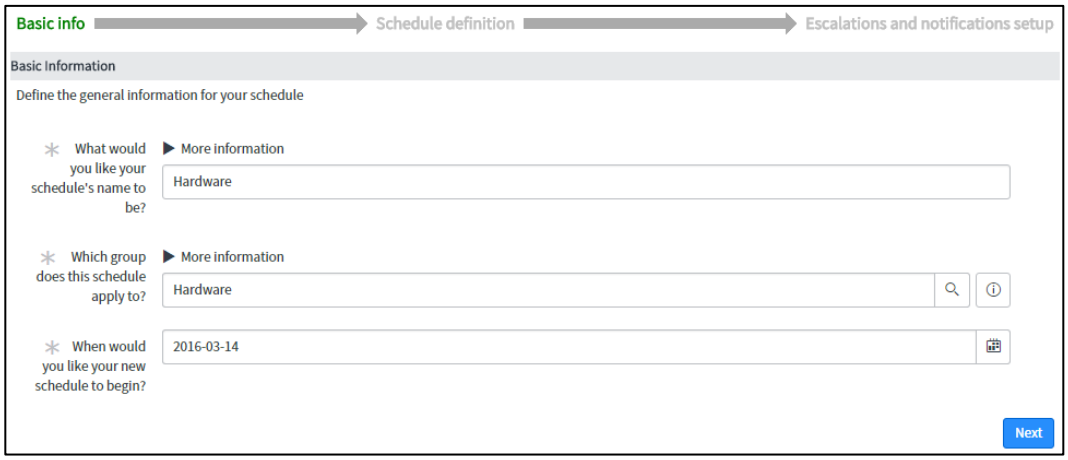
This lab explains how to create a new On-Call Rotation (Rota) and trigger rule with Notify using bidirectional SMS capabilities.

Lab 6.0 Create On-Call Rota

Create New On-Call Rota

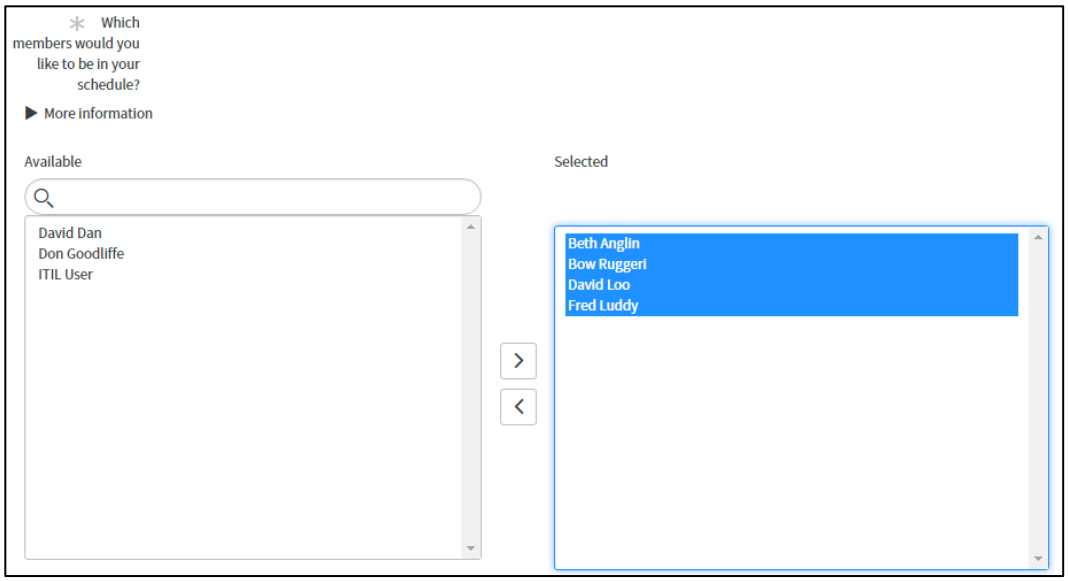
1. In the Filter navigator box, type **on-call**.
2. Click **Create New Schedule** under Administration.
3. Enter the following information:

   What would you like your schedule's name to be?: **Hardware**
   Which group does this schedule apply to?: **Hardware**



4. Click **Next**.
5. Choose **Yes** to use an existing schedule and choose **24 x 7** for the schedule; if not, set by default. Then click **Next**.
6. Choose **1** for the first question on the number of duty rotations.

7. Notice you are able to choose which members of the Hardware team are on the on-call rotation, versus the entire group. Select the following members for the schedule: Beth Anglin, Bow Ruggeri, David Loo, and Fred Luddy. Double-click each member to move them from **Collection** (left side) to **List** (right side).



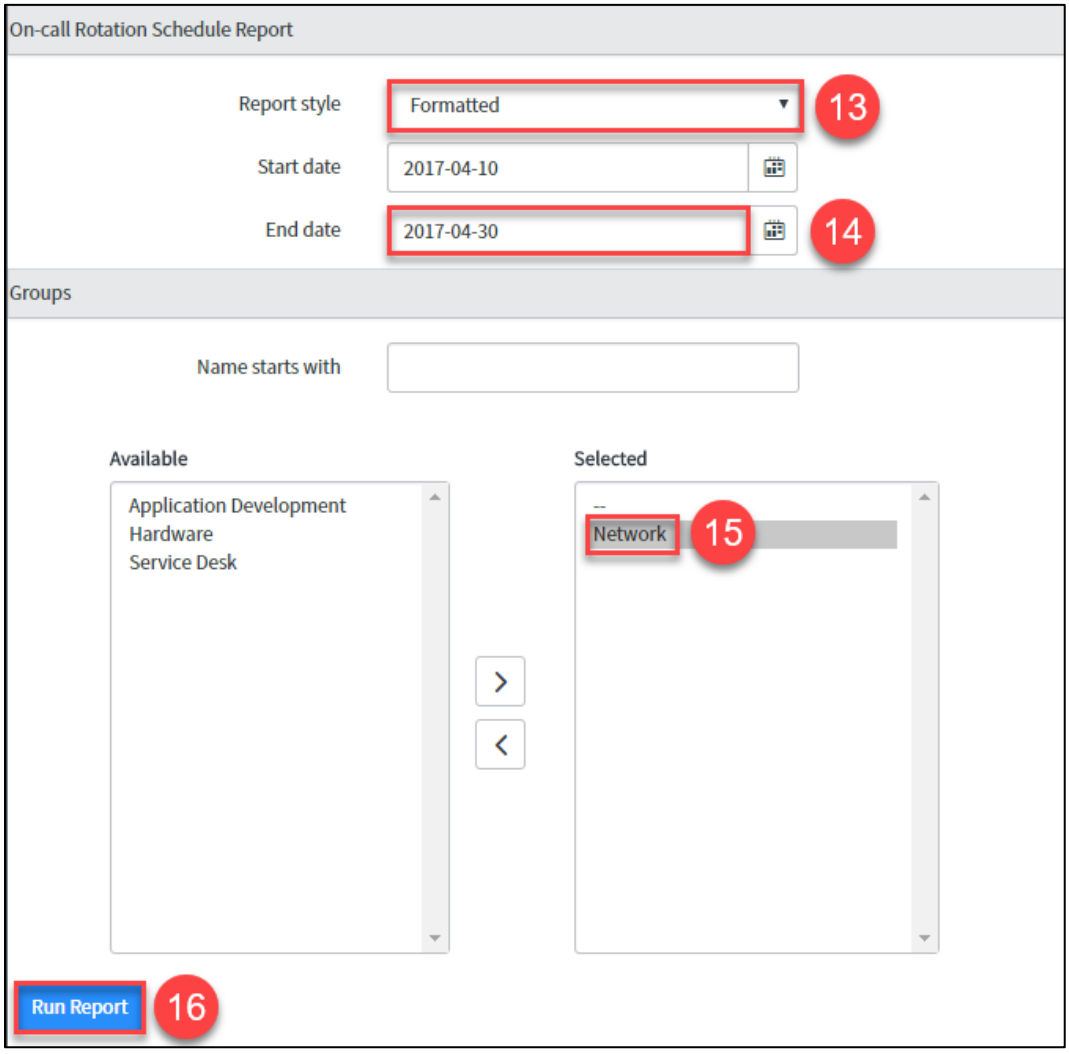
8. Choose **1** for the number of reminders.

9. Choose **15** minutes for the wait until sending the next reminder.

10. Click **Next**.

11. You should now see a calendar view of the Rota. Notice that Beth is on-call starting today for the next week. Bow Ruggeri is on-call after that.

12. There is also another view of the schedule. In the On-Call scheduling application, click **Schedule Report**.



13. There are a few options for report style. Choose **Formatted**.

14. Leave the **Start Date** the default of **Today's Date**, but change the **End Date** to the **end of the month.**

15. Double-click on **Network** to move it to the **Selected** list.

16. Click **Run Report** at the bottom.

17. Feel free to repeat **Steps 12-16** to view the **Table** view of this report or look at another Rota.

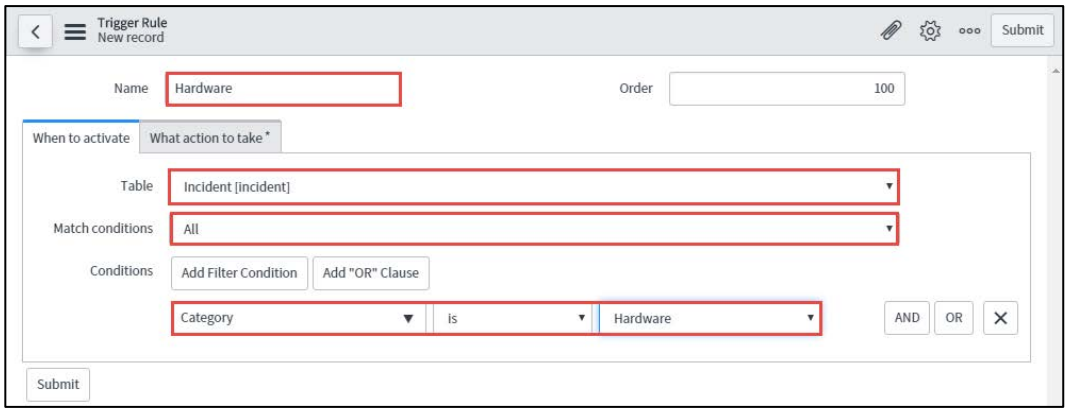
18. Create a trigger rule to assign the Hardware team to a task and start the On-Call workflow.

19. In the On-Call scheduling application, click **Trigger Rules**.

20. Click the **New** button at the top of the list to create a new rule.

21. Enter the following information:

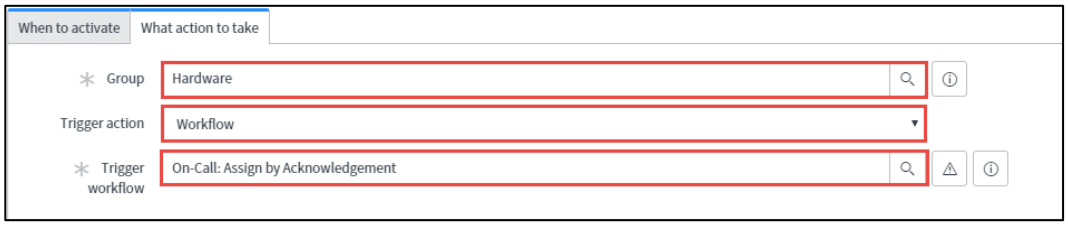
    Name: **Hardware**
    Table: **Incident**
    Match conditions: **All**
    Conditions: **Category is Hardware**



22. Go to the **What action to take** section or click the **What action to take** tab.

23. Enter the following information:

    Group: **Hardware**
    Trigger action: **Workflow**
    Trigger workflow: **On-Call: Assign by Acknowledgement**



24. Click **Submit**.

Lab Success Verification

1. A new Rota and Trigger Rule for the Hardware Team has been created.

Lab Goal

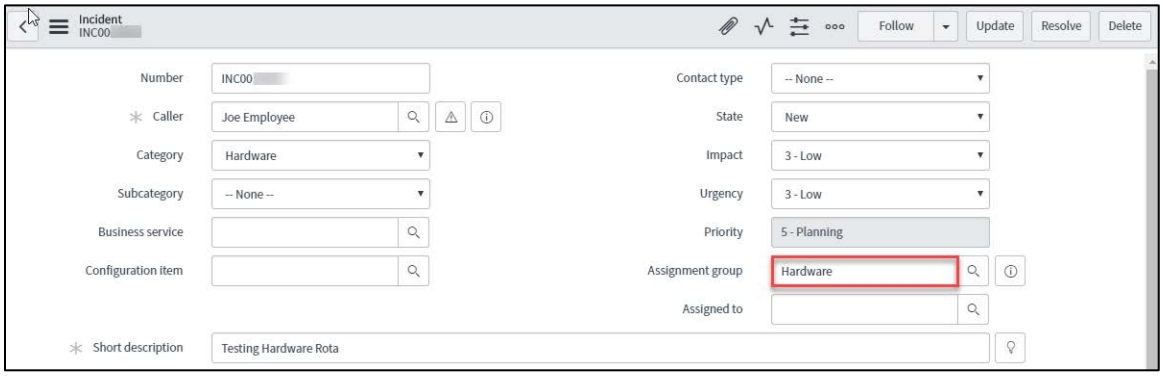
Lab 6.1 Test Hardware Rota

This lab explains how to test a Hardware Rota.

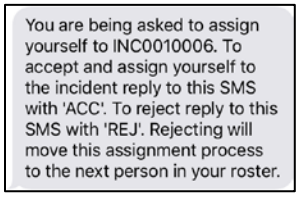
After you create a Hardware Rota and Trigger Rule, any new incident in which the category is Hardware automatically is assigned to the Hardware team and a workflow starts to assign the On-Call person through a verification process.

Create Incident to Test Hardware Rota

1. In the Filter navigator box, type **incident**.
2. Click **Create New** under the Incident application.
3. Enter **Joe Employee** in the **Caller** field, Change the **Category** to **Hardware** and enter **Testing Hardware Rota** in the **Short Description** field. Note that the **Assignment** group is currently empty as well as the **Assigned to** person.
4. Click **Additional actions** and select **Save** to remain on this record.
5. Note the Incident number, which is used in later steps.
6. After the record has been saved, the Assignment group should automatically populate with **Hardware**. This is because of the trigger rule you created.



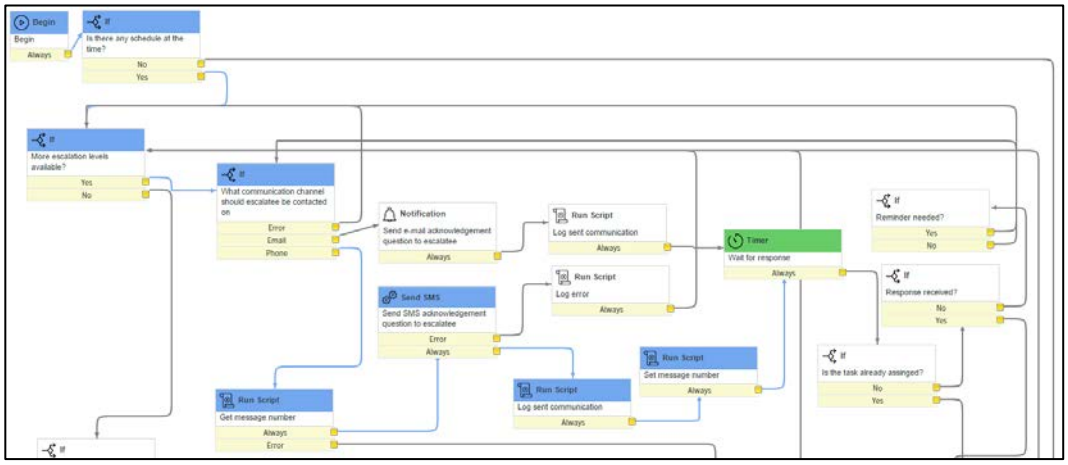
7. You should receive a Text on your Mobile Phone momentarily, asking you to accept or reject the assignment of this new incident.



8. Before you respond to this text, view the workflow. Click **Additional actions** and select **Reload form** to refresh the record. Scroll down to Related Links. Click **Show Workflow**.

   Note: Trigger rules are event-based. You may need to reload the page a few times before the Show Workflow related link shows up.

9. This is an out-of-the-box workflow that comes with the new version of On-Call Rotation. Notice that the workflow is waiting for a response from the SMS that was generated.



10. Respond back to the SMS that you received earlier. Reply with **ACC** to accept the assignment.

11. Go back to the Incident. If you navigated away from the incident, in the filter navigator box, type **incident**. Click **Open** under the Incident application. In the list to the right, open the incident previously created. The number is in the SMS you received.

12. Notice that Beth Anglin is now assigned to the incident. If you recall, you added your phone number to her user record earlier.

Lab Success Verification

1. Incidents of category Hardware are automatically assigned to the Hardware Group.
2. The On-Call person receives an SMS message to accept or reject the assignment.
3. Replying to the SMS updates the Incident.

Lab Goal

Lab 7.0
Call Center for Incident Management

Out-of-the-box ServiceNow includes a workflow called Call Center for Incident Management with the CTI (com.snc.cti) plugin. This is an example workflow allowing customers to run an IVR call center in ServiceNow utilizing Notify and OpenFrame.

Note: WebRTC is only supported in **Chrome** and **Firefox**. If you do not have one of these browsers, you can skip this lab.

This lab introduces this workflow and explains how to configure it to alert callers of open Priority 1 incidents. Note that this workflow leverages a feature included in the CTI plugin called Matching Rules, which you can access via the System Policy\Rules application. The matching rules define who to route incoming calls to based on conditions. Out of the box, all incidents are routed to members of the Phone Call Queue group. Your lab instance was set up to make System Administrator a member of this group.

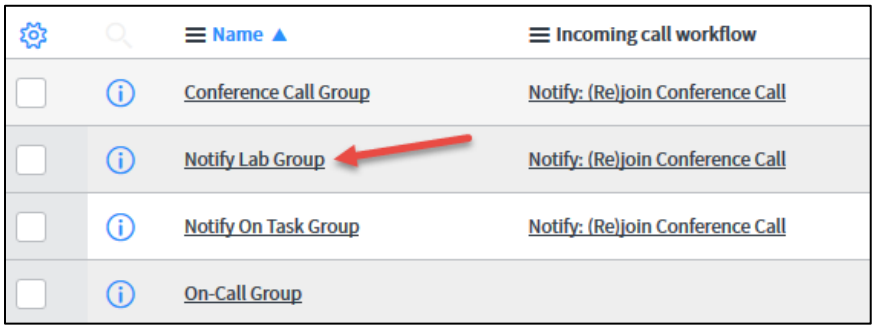
Set Up and Test the Call Center for Incident Management Workflow

1. Lab 1.0 discussed the Number Group controls that workflow executes when a call is received to a phone number. When we set up the Notify Lab Number Group, we chose an out-of-the-box workflow called **Notify: (Re)join Conference Call** to process incoming calls. This workflow allows callers to join conference calls. Now, change the number group to use the **Call Center for Incident Management** workflow.

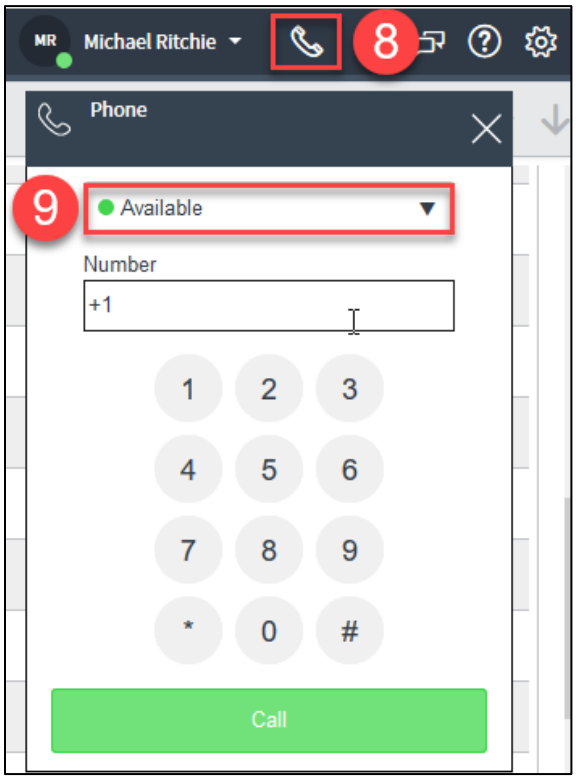
2. In the Filter navigator box, type **notify**.

3. Click **Number Groups** under the Notify Application.

4. Click the **Notify Lab Group** row to edit it.

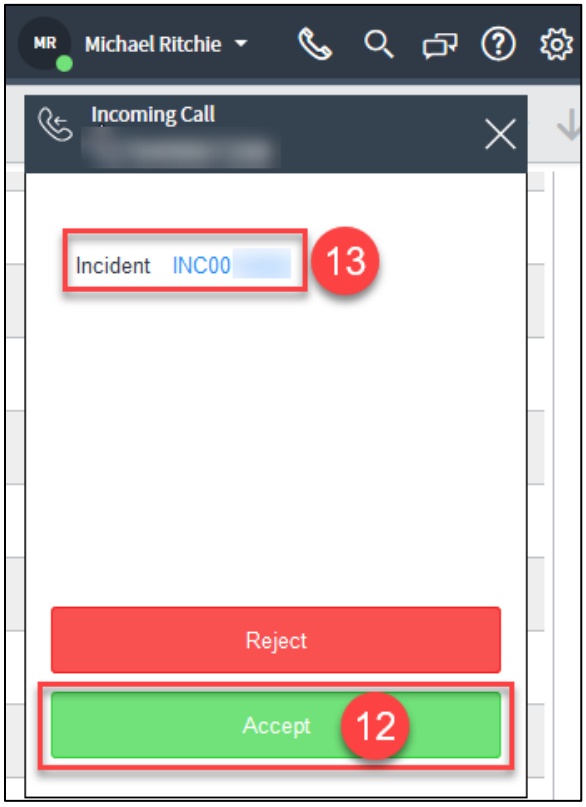


5. Change the Incoming call workflow to **Call Center for Incident Management**.
6. Click **Update**.
7. Now let's try it!
8. Make sure you are logged into your ServiceNow instance in either Chrome or Firefox. As previously mentioned, this workflow routes calls to available agents; therefore, you must be set to "available" status. In the top right corner, click the phone icon.



9. At the top of the phone dialog, change the status from Wrap Up to **Available**.
10. Call your instance Notify number. This is the same number that sent several SMS messages in previous labs.
11. At the first prompt, enter **2** to create a new incident.

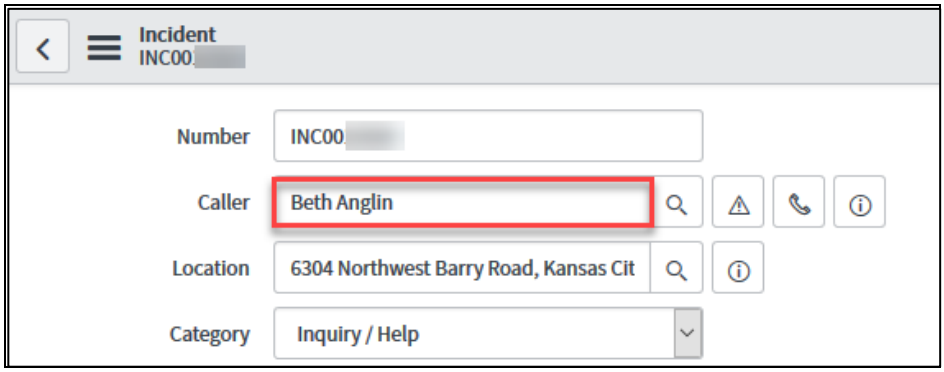
12. Shortly, your computer browser will ring and the phone dialog will change, allowing you to accept or reject the call. Click **Accept**.



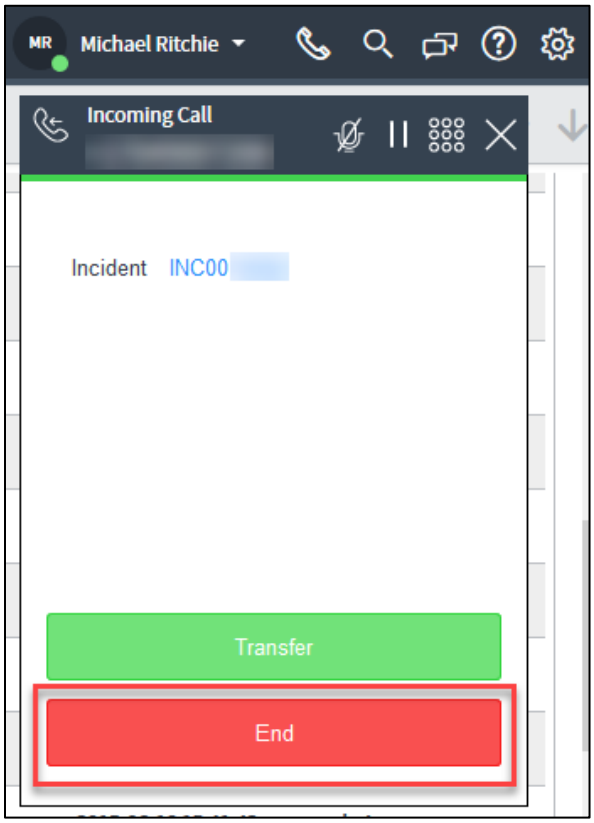
13. Included in the dialog is a link to the incident that was generated by the workflow. Click the **incident number** to be taken to that incident.

14. Notice that Beth Anglin was automatically set as the caller of the incident based on caller ID matching.

    Note: You could have Joe Employee as your caller since both of those users were updated with your phone numbers in Lab 1.1.



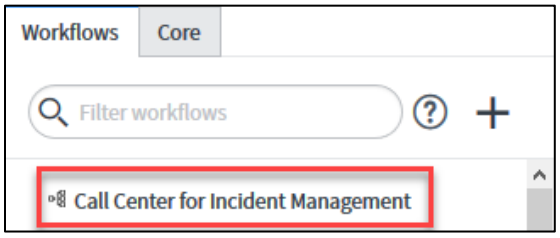
15. In the phone dialog, click **End** to end the call.



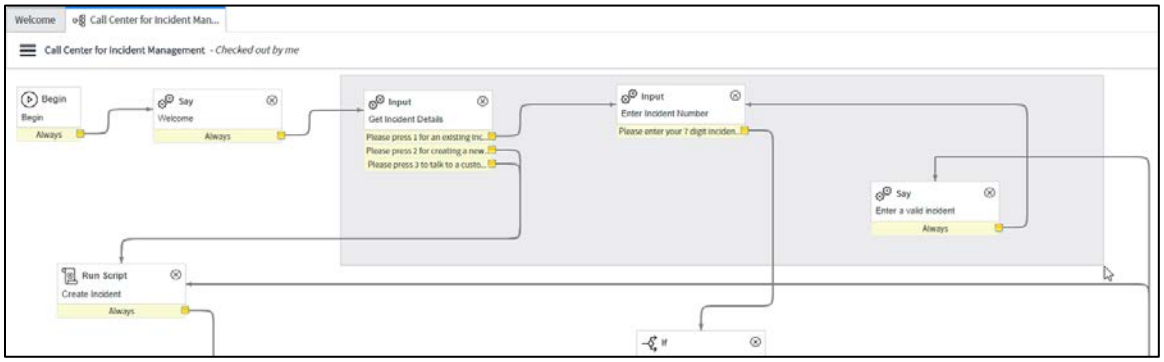
Introduce Priority 1 Incidents to Callers

Many call centers are set up to inform users of any high-priority or severity issues before allowing them to open a new incident. The following steps implement that use case into this out-of-the-box workflow.

1. In the Filter navigator box, type **workflow**.
2. Click **Workflow Editor** in the Workflow Application. Another browser tab appears.
3. On the right, click **Call Center for Incident Management** at the top of the Workflows tab to edit it.



4. Click **Workflow Actions** and select **Checkout**.
5. Before allowing the caller to create an incident, check to see if there is an open P1 incident. Move the activities after the **Say (Welcome)** activity to the right to add a few additional activities. Using the mouse, highlight the **Input (Get Incident Details)**, **Input (Enter Incident Number)**, and **Say (Enter a valid Incident)** activities and then release the mouse button. The three activities should turn blue.



6. Put the mouse over any of the three highlighted activities and move them to the right.
7. On the right side of the screen, click the **Core** tab. This presents activities that can be leveraged in the workflow.

8. Under Core Activities\Conditions, left mouse click **If** and drag it over the line between **Say (Welcome)** and **Input (Get Incident Details)** until the line turns blue. Release the left mouse button.

9. Enter the following information:

   Name: **Check for P1 Incidents**
   Advanced: **checked**
   Script:

```
answer = ifScript();

function ifScript() {
   var incidentRec = new GlideRecord("incident");
   // Query for open/active incidents that are Priority 1 that have been
opened in the last 15 minutes
   incidentRec.addEncodedQuery("active=true^priority=1^opened_atONLast 15
minutes@javascript:gs.minutesAgoStart(15)@javascript:gs.minutesAgoEnd(0)")
;
   incidentRec.query();
   if (incidentRec.next()) {
         // Put together the incident details that we want to share with
the caller
         var taskDetails = "There is currently a priority 1 incident
affecting " + incidentRec.cmdb_ci.getDisplayValue() + ".";
         taskDetails = taskDetails + "Incident description is " +
incidentRec.short_description + ".";
         taskDetails = taskDetails + "The incident is currently being
worked on by the  " + incidentRec.assignment_group.getDisplayValue() + "
team.";
         taskDetails = taskDetails + "Last update was at" +
incidentRec.sys_updated_on.getDisplayValue();
         // Pass the incident details to a workflow variable so that the
next Say activity can repeat these details.
         workflow.scratchpad.p1_text = taskDetails;
         return 'yes';
   }
   return 'no';
}
```

   Note: The script can also be copied from the Lab Management application under the Lab Exercises section by clicking **Lab 7 Call Center for IM Scripts**.

10. Click **Submit**.

11. Next, if there is a P1 incident, provide status details. For this example, read the incident configuration item, short description, assignment group, and last updated timestamp.

12. Under Core Activities\Notify, left mouse click **Say** and drag it over the line between **If Yes** Condition and the **Input (Get Incident Details)** activity until the line turns blue. Release the left mouse button.

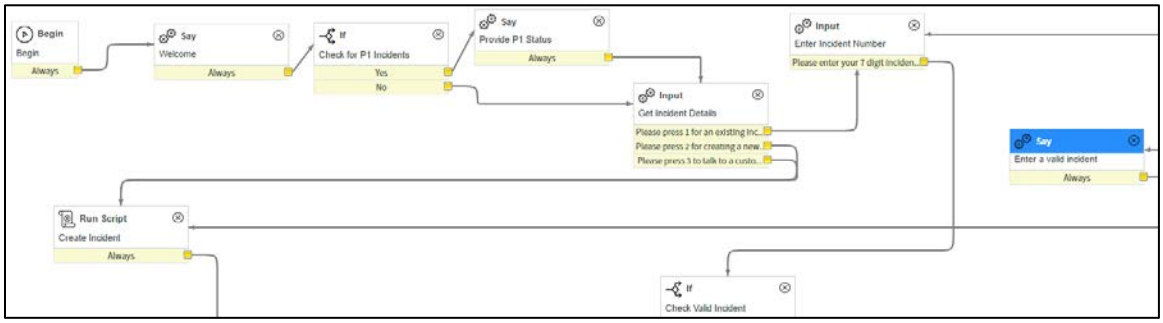
13. Enter the following information:

    Name: **Provide P1 Status**
    Text to say: **${p1_text}**

14. Click **Submit**.

15. Connect the **No** condition on the If Activity to the **Input** (**Get Incident Details**) activity.

16. Your workflow should look similar to the following:



17. Click **Workflow Actions** and select **Publish**.

18. Now, test it!

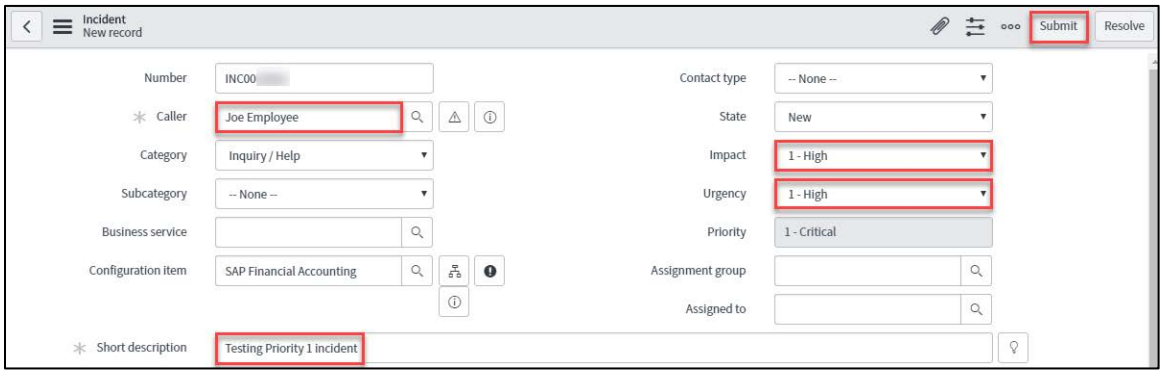
19. Call your instance Notify number. This is the same number that sent several SMS messages in previous labs. Since no P1 incident is open, you are prompted to Press 1 for an existing incident and so on.

20. **Hang up** the call.

21. In the Filter navigator box, type **incident**.

22. Click **Create New** under the Incident application.

23. Change the **Caller** to **Joe Employee**, **Configuration Item** to **SAP Financial Accounting, Impact** to **1-High**, **Urgency** to **1-High**, and enter **Testing Priority 1 incident** in the **Short Description** field.



24. Click **Submit**.

25. Call your instance Notify number.

26. Now that you have a recent Priority 1 incident, the details of the new incident should be dictated before the you can Press 1 for an existing incident.

27. **Hang up** the call.

Lab Success Verification

1. Calls can be accepted via your Chrome or Firefox browser.

2. Details of a Priority 1 incident are dictated to the caller if one is open.

Lab Goal

This lab explains how to create a workflow to provide Incident and Problem status.

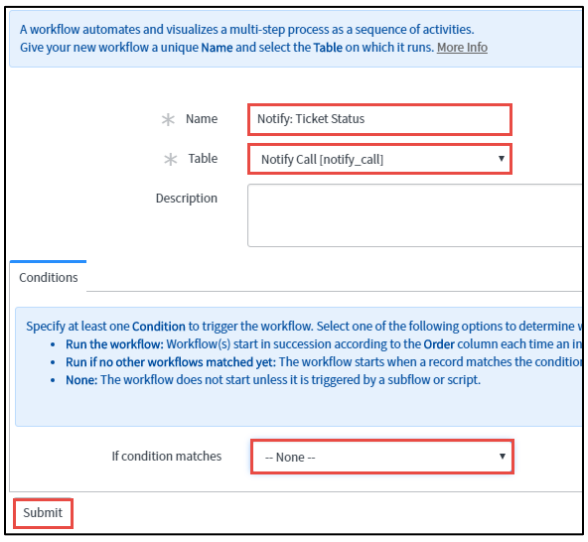
The Notify Text-to-Voice capabilities allow you to permit users to call and get a status of tasks within ServiceNow.

Lab 8.0 Ticket Status via Voice

Create a New Ticket Status Workflow

1. In the Filter navigator box, type **workflow**.
2. Click **Workflow Editor** in the Workflow Application. Another browser tab appears.
3. On the right, click the **+** icon on the Workflows tab to create a new workflow.
4. Enter the following information:

   Name: **Notify: Ticket Status**
   Table: **Notify Call [notify_call]**
   If condition matches: **None**



5. Click **Submit**.

6. You should move to a workflow with just a **Begin** and **End**.

7. Move the **End** activity on the right side of the screen by left-clicking on it.

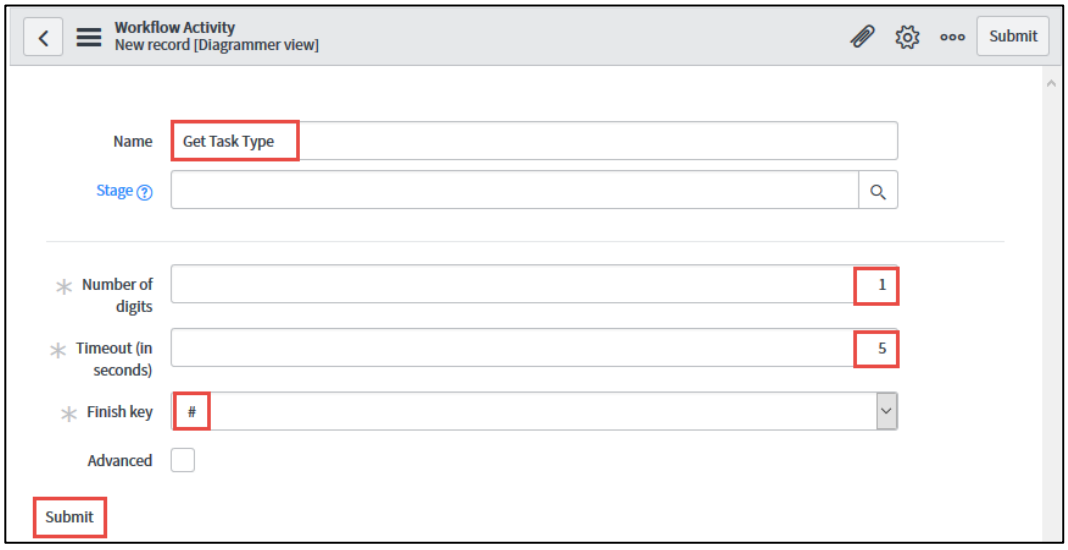
8. First, get input from the user on the type of ticket on which to check status. This is done via an Input Workflow Activity. By default, the text that Twilio reads to the user comes from the Input Activity Condition Name. This can also be scripted to be more dynamic.

9. On the right side of the screen, click the **Core** tab. This presents activities that can be leveraged in the workflow.

10. Under Core Activities\Notify, left mouse click **Input** and drag it over the line between **Begin** and **End** until the line turns blue. Release the mouse button.

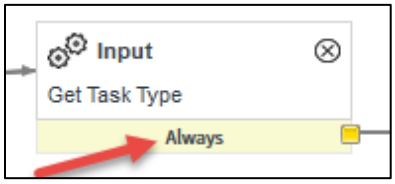
11. Enter the following information:

    Name: **Get Task Type**
    Number of digits: **1** – default value
    Timeout: **5** – default value
    Finish key: # – default value



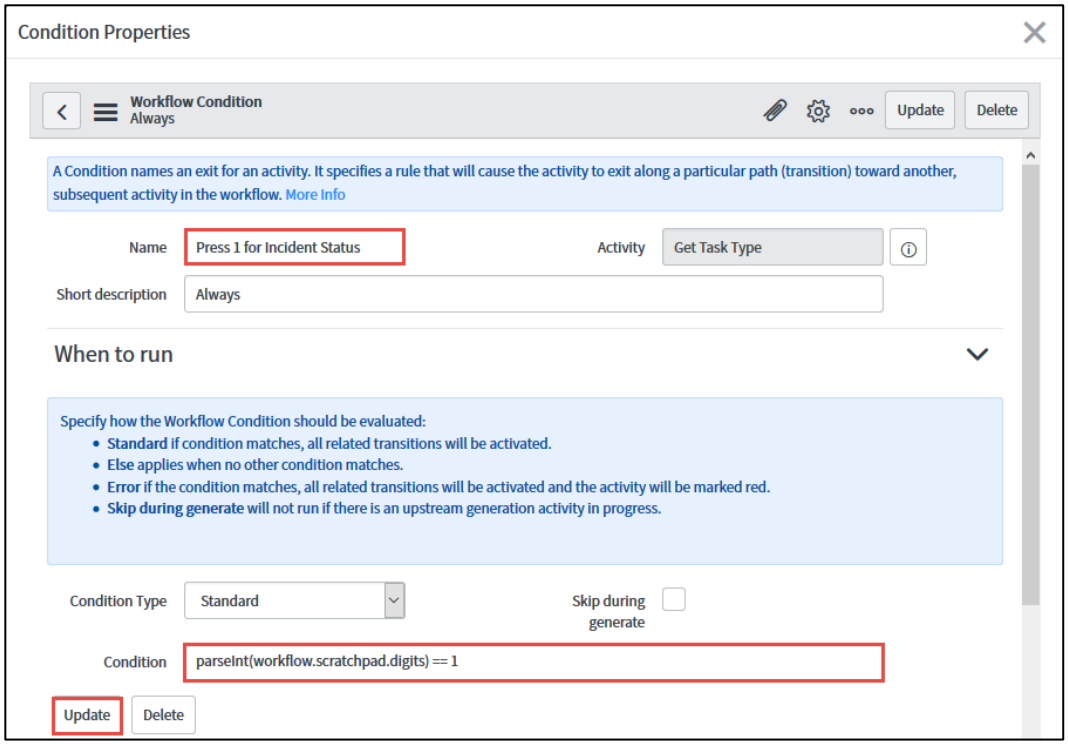
12. Click **Submit**.

13. Now, ask the user to press 1 for incident. Double-click on **Always** in the yellow box at the bottom of the Input activity.



14. A Condition Properties dialog appears. Change the following values:

    Name: **Press 1 for Incident Status**
    Condition: **parseInt(workflow.scratchpad.digits) == 1**



15. Click **Update**.

16. Right-click the **Input** activity above the yellow box and choose **Add Condition**.

17. A New Workflow Condition dialog appears. Enter the following values:

    Name: **Press 2 for Problem Status**
    Condition: **parseInt(workflow.scratchpad.digits) == 2**

18. Click **Submit**.

19. When the caller enters a choice, the Input workflow activity stores that value in a Workflow Scratchpad variable called workflow.scratchpad.digits. Capture that value for use later in the workflow since we have another input activity to prompt for a ticket number and the ticket type value will be overwritten.

20. Under Core Activities\Utilities, left-click on **Run Script** and drag it over the line between **Input** and **End** until the line turns blue. Release the mouse button.

21. Enter the following information:

    Name: **Store Task Type**
    Script:

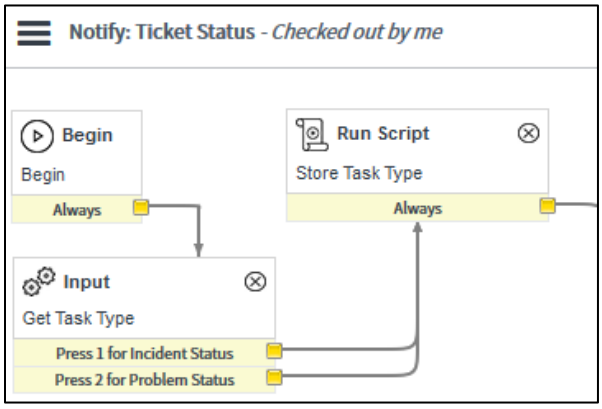
```
workflow.scratchpad.tickettype = workflow.scratchpad.digits;
```

    Note: The script can also be copied from the Lab Management application under the Lab Exercises section by clicking **Lab 8 Ticket Status via Voice Scripts**.

22. Click **Submit**.

23. Connect the **Press 2 for Problem Status** condition on the Input Activity to the **Run Script** activity.

24. Your workflow should look similar to the example.



25. Next, prompt for the ticket number. Under Core Activities\Notify, left-click on **Input** and drag it over the line between **Run Script** and **End** until the line turns blue. Release the mouse button.

26. Enter the following information:

    Name: **Get Ticket Number**
    Number of digits: **7**
    Timeout: **5** – default value
    Finish key: # – default value

27. Click **Submit**.

28. Now, ask the user to enter the ticket number. Double-click on **Always** in the yellow box at the bottom of the Input activity.

29. Enter the following information:

    Name: **Please enter the 7 digit ticket number. Please don't include the letter prefix.**
    Condition: **true** – default value

30. Click **Update**.

31. Next, determine if the caller entered a valid Incident or Problem. Also, capture the task **GlideRecord** in a workflow scratchpad variable to use in a later workflow activity.

32. Under Core Activities\Conditions, left-click on **If** and drag it over the line between **Input** (Get Ticket Number) and **End** until the line turns blue. Release the mouse button.

33. Enter the following information:

    Name: **Valid record number entered?**
    Advanced: **checked**
    Script:

```
if (parseInt(workflow.scratchpad.tickettype) == 1) {
   answer = isValidTask("incident");
} else if (parseInt(workflow.scratchpad.tickettype) == 2) {
   answer = isValidTask("problem");
} else {
   answer = 'no';
}

function isValidTask(taskTable) {
    // get digits from the input activity
    var ticketNum = workflow.scratchpad.digits;

    // get task that matches
    var taskRec = new GlideRecord(taskTable);
    taskRec.addQuery('number', 'CONTAINS', ticketNum.toString());
   taskRec.query();

    // is there a record?
    if (taskRec.next()) {
        // store the task GlideRecord for use with ticket status
        workflow.scratchpad.task = taskRec;
        return 'yes';
    }

    return 'no';
}
```

    Note: The script can also be copied from the Lab Management application under the Lab Exercises section by clicking **Lab 8 Ticket Status via Voice Scripts**.

34. Click **Submit**.

35. Next, if the caller entered a valid task number, you need to provide status details. For this example, read the task number, assignment group, and last updated timestamp.

36. Under Core Activities\Notify, left-click on **Say** and drag it over the line between **If Yes** Condition and the **End** until the line turns blue. Release the mouse button.

37. Enter the following information:

    Name: **Provide Ticket Status**
    Advanced: **checked**
    Script:

```
// generate the text to say and the language to say it in
text = generateTextToSay();

/**
 * Generate the text to say and the language to say it in
 *
 * @returns {{text: string, language: string}}
 */
function generateTextToSay() {
    // construct the result object
   var taskRec = workflow.scratchpad.task;
   var taskDetails = taskRec.sys_class_name + " " + taskRec.number + " is
currently " + taskRec.state.getDisplayValue();
   taskDetails = taskDetails + " and assigned to the " +
taskRec.assignment_group.getDisplayValue() + " team.";
   taskDetails = taskDetails + "Last update was at" +
taskRec.sys_updated_on.getDisplayValue();
    //var taskDetails = taskRec;
   var result = {
        // define the text to say
        text        : taskDetails,

        // and the language to say it in (see
        // or query the notify_language table
        // for supported languages)
        language    : 'en-US'
    };

    // return the result
    return result;
}
```

    Note: The script can also be copied from the Lab Management application under the Lab Exercises section by clicking **Lab 8 Ticket Status via Voice Scripts**.

38. Click **Submit**.

39. Next, if the caller did not enter a valid task number, notify him/her and end the call.

40. Under Core Activities\Notify, left-click on **Say** and drag it below the existing Say activity. Release the left button.

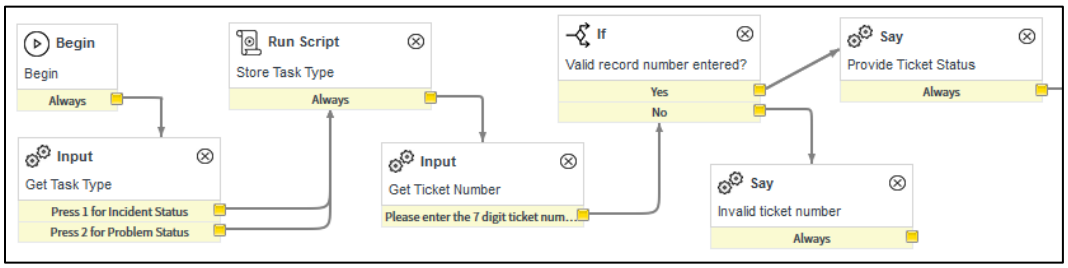
41. Enter the following information:

    Name: **Invalid ticket number**
    Text to say: **The ticket number you entered is not valid.**

42. Click **Submit**.

43. Connect the **No** condition on the If Activity to the **Say** (Invalid ticket number) activity.

44. The workflow should look similar to the example.



45. Next, ask the caller whether to repeat the ticket status or not.

46. Under Core Activities\Notify, left-click on **Input** and drag it over the line between **Say** (Provide Ticket Status) and the **End** until the line turns blue. Release the mouse button.

47. Enter the following information:

    Name: **Ask if Repeat**
    Number of digits: **1** – default value
    Timeout: **5** – default value
    Finish key: # – default value

48. Click **Submit**.

49. Now, ask the user to press **1** to repeat. Double-click on **Always** in the yellow box at the bottom of the Input activity.

50. A Condition Properties dialog appears. Change the following values:

    Name: **Press 1 to Repeat**
    Condition: **parseInt(workflow.scratchpad.digits) == 1**

51. Click **Update**.

52. Right-click on the **Input** (Ask if Repeat) activity, above the yellow box, and choose **Add Condition**.

53. A New Workflow Condition dialog appears. Enter the following values:

    Name: **Press 2 to Hang Up**
    Condition: **parseInt(workflow.scratchpad.digits) != 1**

54. Click **Submit**.

55. Click the line coming from the **Press 1 to Repeat** condition on the Input (Ask if Repeat) activity so it turns blue. Press the delete button on the keyboard to remove it.

56. Connect the **Press 1 to Repeat** condition to the **Say** (Provide Ticket Status) activity.

57. Next, be polite and say **Goodbye** before ending the call.

58. Under Core Activities\Notify, left-click on **Say** and drag it below the **Ask if Repeat** Input activity. Release the left - button.

59. Enter the following information:

    Name: **Good bye**
    Text to say: **You will now be disconnected. Good bye.**

60. Click **Submit**.

61. Connect the **Press 2 to Hang Up** condition to the **Say** (Good bye) activity.

62. Connect the **Say (Invalid ticket number)** to the **Say (Good bye)** activity.

63. Finally, end the call by telling Twilio to hang up.

64. Under Core Activities\Notify, left-click **Hang Up** and drag it beside the **Say (Good bye)** activity. Release the left button.

65. Enter the following information:

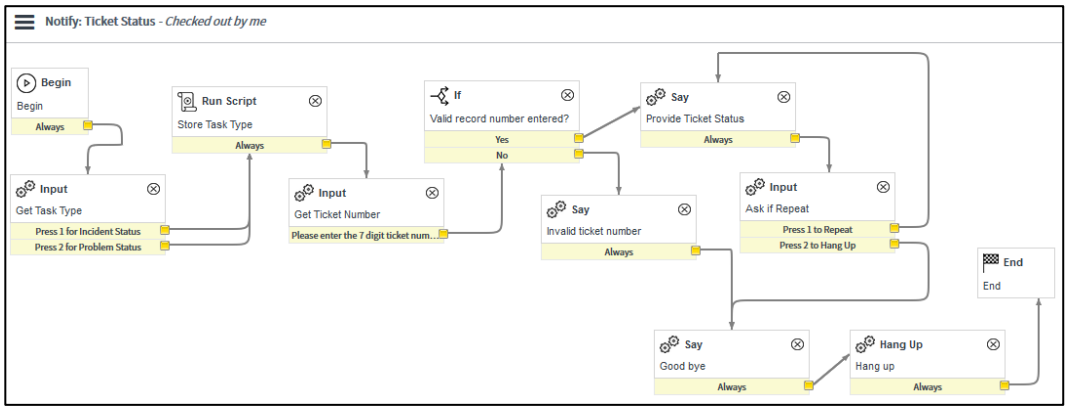
    Name: **Hang Up**

66. Click **Submit**.

67. Connect the **Say (Good bye)** activity to the **Hang Up** activity.

68. Connect the **Hang Up** activity to the **End** activity.

69. Your workflow should look similar to the example.



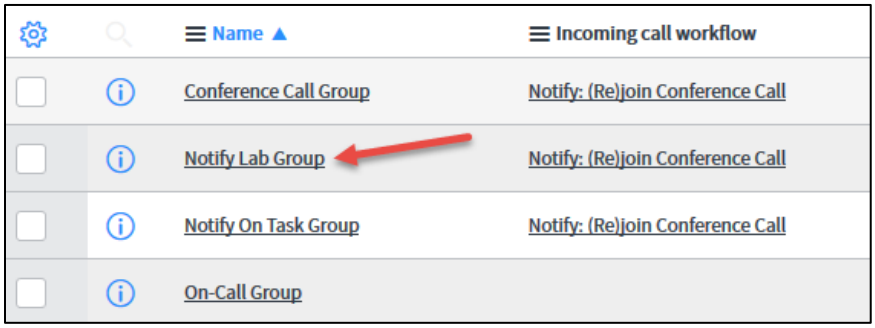
70. Click **Workflow Actions** and select **Publish**.

71. Lab 1.0 discussed that the Number Group controls which workflow executes when a call is received to a phone number. When we set up the Notify Lab Number Group, we chose an out-of-the-box workflow called **Notify: (Re)join Conference Call** to process incoming calls. This workflow allows callers to join conference calls. Now, change the number group to use the new workflow.

72. In the Filter navigator box, type **notify**.

73. Click **Number Groups** under the Notify Application.

74. Click the **Notify Lab Group** row to edit it.



75. Change the Incoming call workflow to **Notify: Ticket Status**.

76. Click **Update**.

77. Now, test it!
78. In the Filter navigator box, type **incident**.
79. Click **Open** under the Incident application.
80. Pick any ticket number in the list to use as the test.
81. Call the Notify number. This is the same number that sent several SMS messages in previous labs.
82. At the first prompt, enter **1** to check status of an Incident ticket.
83. When prompted enter the **5-7** numeric digits of the incident number you choose in **Step 80**.
84. Continue to follow prompts to your liking. Feel free to call back and try a problem number as well.
85. You can also navigate to Workflow\Active Contexts and view the executing workflow.

Lab Success Verification

1. By calling your Twilio number you were able to get the current status of an incident record.

Lab Goal

Lab 9.0 Orchestration with Notify

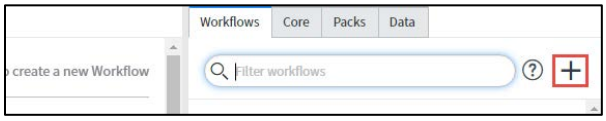
This lab explains how to create a workflow that reaches out to the assignee of an incident via SMS for taking action on the incident by responding to an SMS.

Notify paired with ServiceNow Orchestration is a powerful combination. Many incidents are resolved with the same script execution especially in batch process automation issues.

Note: These lab instances do not have Orchestration turned on; however, we mimic the capabilities in our workflow. Consider what actions to allow your users to take via automation. This lab walks you through the initial setup.

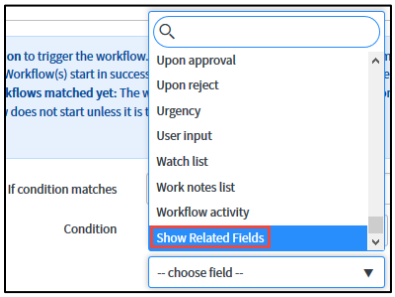
Create a New Workflow

1. In the Filter navigator box, type **workflow**.
2. Click **Workflow Editor** in the Workflow Application. Another browser tab appears.
3. On the right, click the **+** icon on the Workflows tab to create a new workflow.



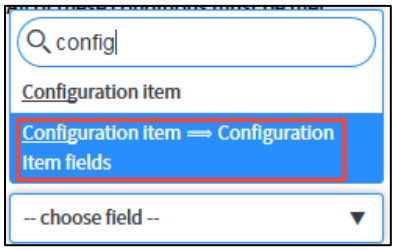
4. Enter the following information:

   Name: **Linux Server Troubleshooting**
   Table: **Incident [incident]**
   If condition matches: **Run the workflow**
   Under condition, click choose **field** and scroll to the bottom and select: **Show Related Fields**

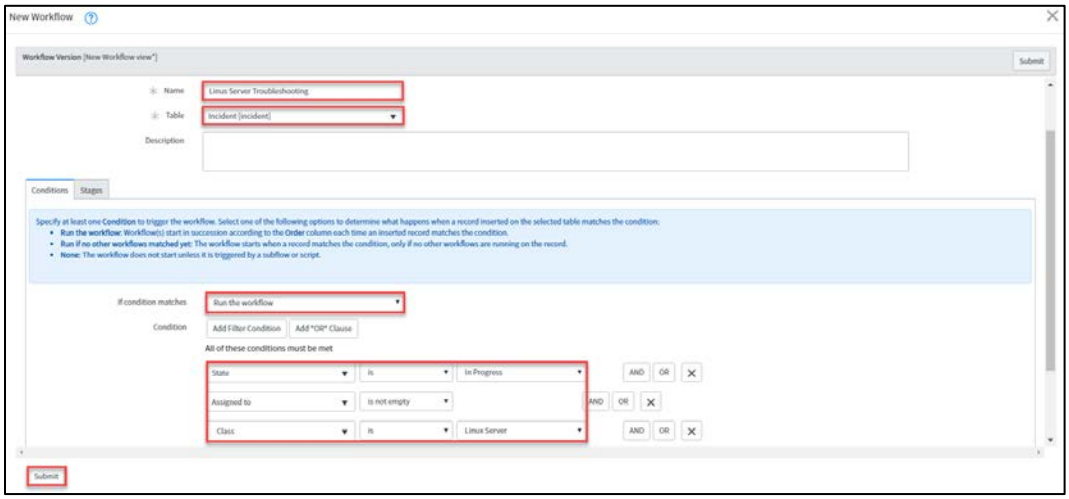


5. Set the following conditions:

   State is **In Progress** – click AND button
   Assigned To **is not empty** – click AND button
   Click Configuration Item ➔ Configuration Item fields



6. After the list refreshes, set **Class is Linux Server**.



7. Click **Submit**.

8. You move to a workflow with just a **Begin** and **End**.

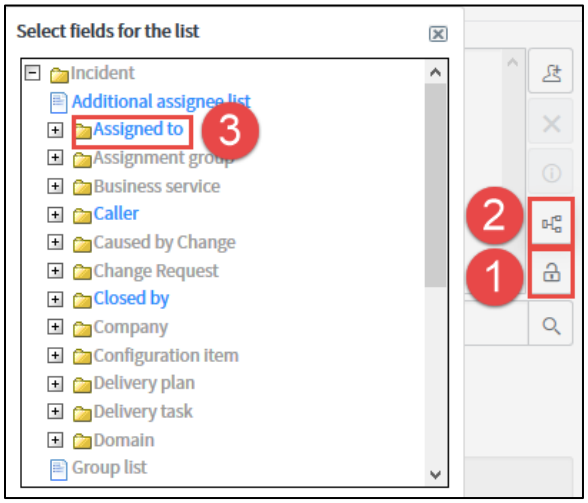
9. Move the **End** activity to the right side of the screen by left-clicking it.

10. On the right side of the screen, click the **Core** tab. This displays activities that can be leveraged in the workflow.

11. Under Core Activities\Notify, find **Send SMS**. Left-click and drag it over the line between **Begin** and **End** until the line turns blue. Release the left button. After the line turns blue, the activity is inserted between those two activities.

12. Enter the following information:

Name: **Send SMS orchestration question to assignee**
From: **Click the Magnifying Glass and choose your Notify Number**
To: **Click the Lock button and then the Select Fields button and choose Assigned To**

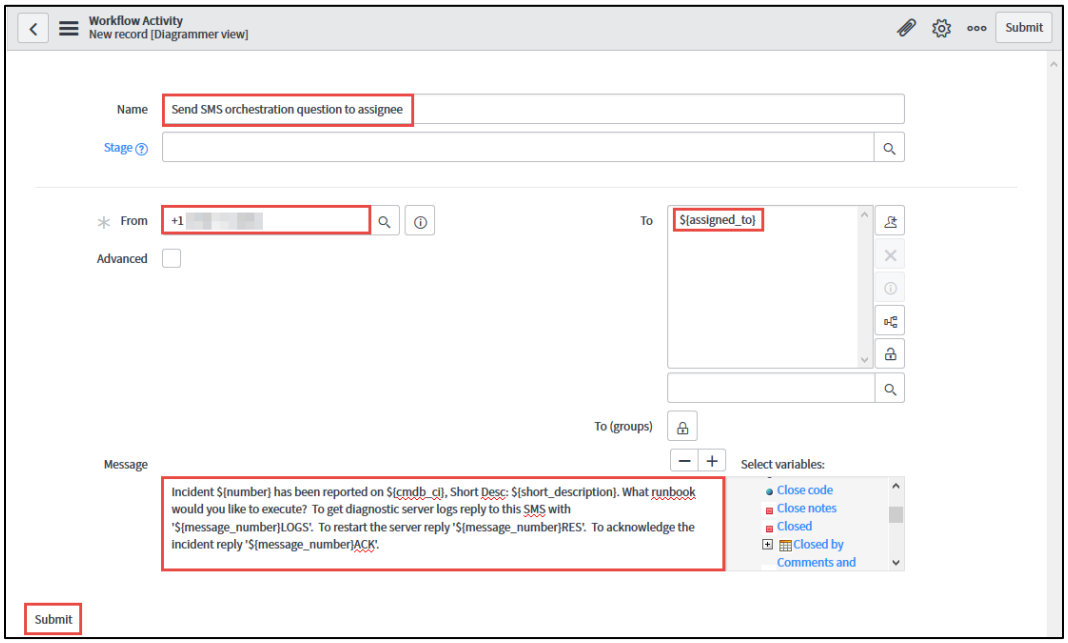


Message:

```
Incident ${number} has been reported on ${cmdb_ci.name}, Short Desc:
${short_description}. What runbook would you like to execute? To get
diagnostic server logs reply to this SMS with 'LOGS'. To restart the server
reply 'RES'. To acknowledge the incident reply 'ACK'.
```

Note: The message above can also be copied from the Lab Management application under the Lab Exercises section by clicking **Lab 9 Orchestration with Notify Scripts**.

13. Your activity should look similar to the example.

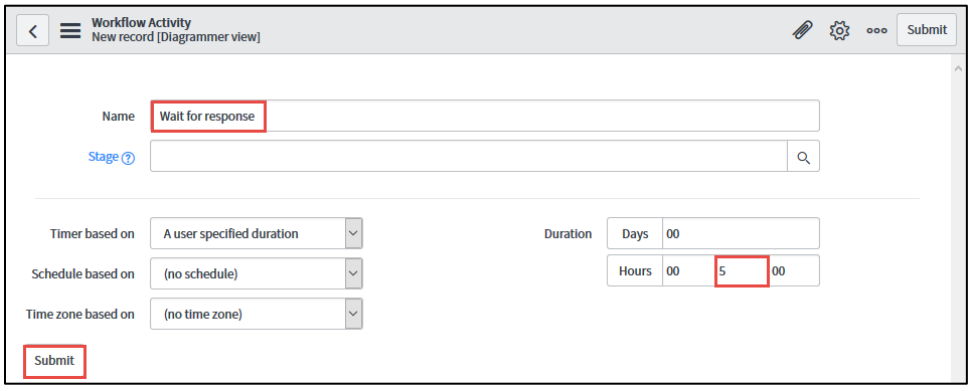


14. Click **Submit**.

15. Under Core Activities\Timers, left-click **Timer** and drag it over the line between Send SMS and End until the line turns blue. Release the left button.

16. Enter the following information:

    Name: **Wait for response**
    Duration: **5 Minutes**



17. Click **Submit**.

18. Under Core Activities\Conditions, left-click on **If** and drag it over the line between **Timer** and **End** until the line turns blue. Release the left button.

19. Enter the following information:

    Name: **Determine if answer received**
    Advanced: **checked**
    Script:

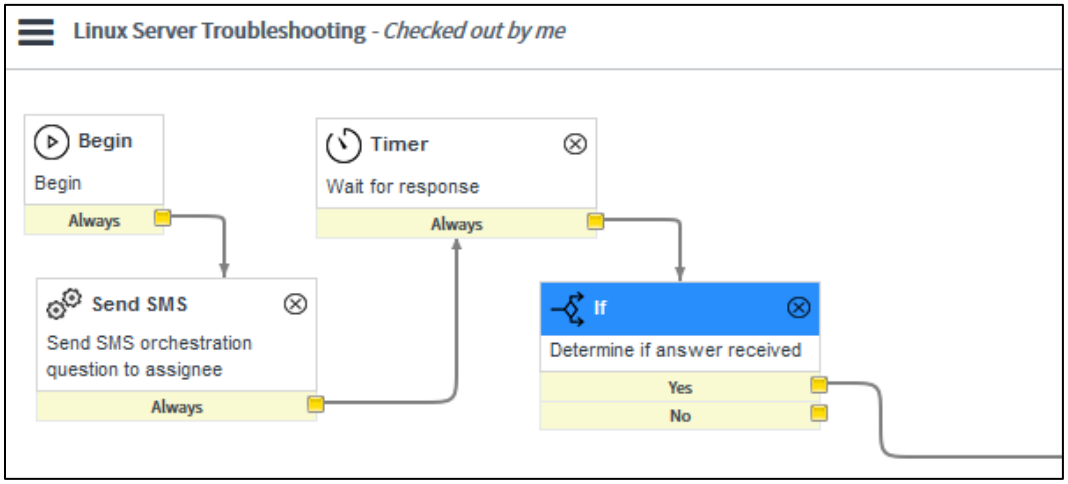
```
answer = ifScript();

function ifScript() {
	//Get Outbound phone number first so we can query for inbound
	var outboundMessage = new GlideRecord("notify_message");
	outboundMessage.addQuery("direction", "outbound");
	outboundMessage.addQuery("table", current.getTableName());
	outboundMessage.addQuery("source", current.sys_id);
	outboundMessage.orderByDesc('sys_created_on');
	outboundMessage.query();
	if (outboundMessage.next()) {
		//Now query for inbound message from same phone number and after the
outbound was created
		var inboundMessage = new GlideRecord("notify_message");
		inboundMessage.addQuery("direction", "inbound");
		inboundMessage.addQuery("phone_number",
outboundMessage.phone_number);
		inboundMessage.addQuery("sys_created_on", ">",
outboundMessage.sys_created_on);
		inboundMessage.orderByDesc('sys_created_on');
		inboundMessage.query();
		if (inboundMessage.next()) {
			var smsResponse = inboundMessage.body;
			//Use Regex to match response to the codes this workflow
expects
			var pattern = new SNC.Regex('/^([0-9]*)(LOGS|RES|ACK)$/i');
			var matches = pattern.match(smsResponse);
			if (matches !== null) {
				//Set a scratchpad variable for use in next activity.
Convert to lower case.
				workflow.scratchpad.response = matches[0].toLowerCase();
				return "yes";
			}
		}
	}
	//If no record is found, return No
	return "no";
}
```

Note: This script can be copied from the Lab Management application under the Lab Exercises section by clicking **Lab 9 Orchestration with Notify Scripts**.

20. Click **Submit**.

21. Your workflow should look similar to the example.



22. Next, determine what action to take based on the code received.

23. Under **Core Activities\Utilities**, left-click **Run Script** and drag it over the line between **If** and **End** until the line turns blue. Release the left button.

24. Enter the following information:

    Name: **Determine Runbook Action**
    Script:

```
//This script will update the Incident work notes with the action taken from
the incident assignee.
//It will also use the Notify API to send a SMS back to the incident assignee
confirming the action taken.
var workNotes = "";
var ackMessage = "";
switch (workflow.scratchpad.response.toLowerCase()) {
      case "logs":
            workNotes = current.assigned_to.name + ' requested diagnostic
logs.';
            ackMessage = "Diagnostic logs have been requested.";
            updateInc(workNotes, ackMessage);
      break;
      case "res":
            workNotes = current.assigned_to.name + ' requested to restart the
server.';
            ackMessage = "Server restart has been requested.";
            updateInc(workNotes, ackMessage);
      break;
      case "ack":
            workNotes = current.assigned_to.name + ' ackowledged, no action
taken.';
            ackMessage = "Incident acknowledged no action taken.";
            updateInc(workNotes, ackMessage);
      break;
}

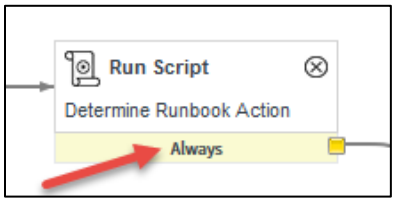
function updateInc(workNotes, ackMessage) {
      current.work_notes = workNotes;

      //The sendSMS function requires a Notify number, use the API to get the
list of available numbers and use the first one.
      var notify = new SNC.Notify();
      var notifyNumber = notify.getPhoneNumbers();
      notifyNumber = notifyNumber.get(0);
      notifyNumber = notifyNumber.getNumber();
      var toNumber = current.assigned_to.mobile_phone;
      notify.sendSMS(notifyNumber, toNumber, ackMessage, current);
}
```

    Note: The script above can also be copied from the Lab Management application under the Lab Exercises section by clicking **Lab 9 Orchestration with Notify Scripts**.

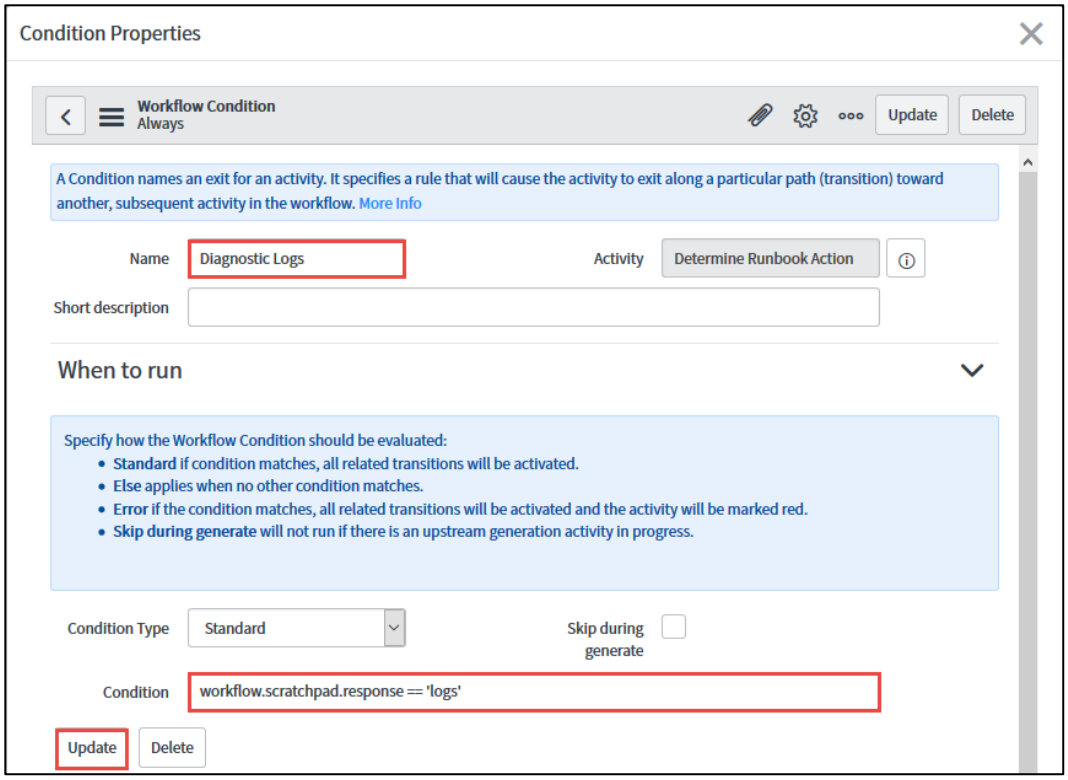
25. Click **Submit**.

26. Double-click on **Always** in the yellow box at the bottom of the Run Script activity.



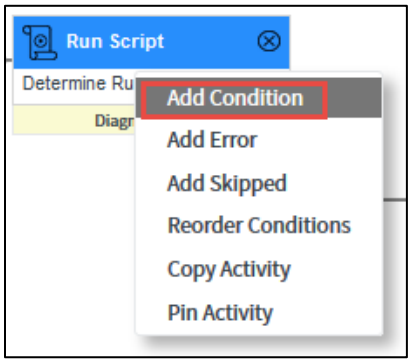
27. A Condition Properties dialog appears. Change the following values:

    Name: **Diagnostic Logs**
    Condition: **workflow.scratchpad.response == 'logs'**



28. Click **Update**.

29. Right-click on the Run Script activity, above the yellow box, and choose **Add Condition**.



30. A New Workflow Condition dialog appears. Enter the following values:

    Name: **Restart Server**
    Condition: **workflow.scratchpad.response == 'res'**

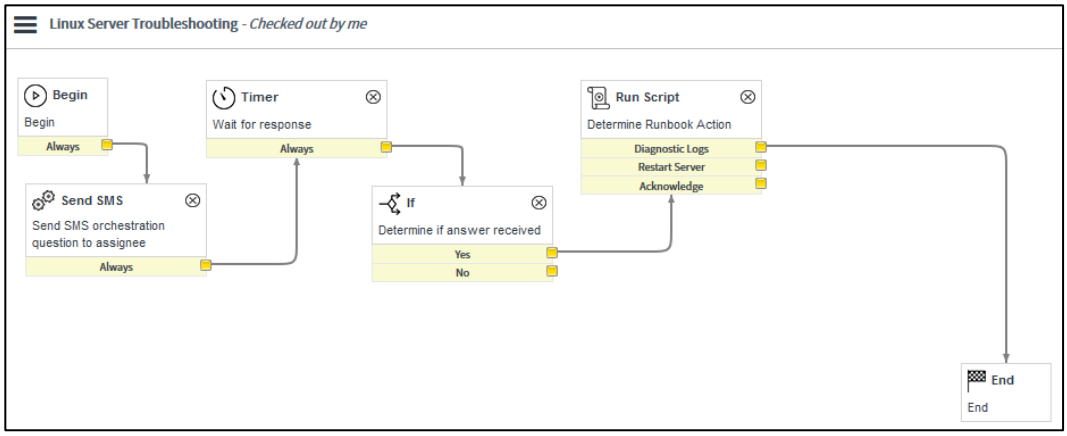
31. Click **Submit**.

32. **Right-click** the Run Script activity, above the yellow box, and choose **Add Condition**. Enter the following values:

    Name: **Acknowledge**
    Condition: **workflow.scratchpad.response == 'ack'**

33. Click **Submit**.

34. Your workflow should look similar to the example.



35. ServiceNow Orchestration is basically custom workflows that integrate with external systems via workflow leveraging Web Services (SOAP and REST), Windows PowerShell, Shell Script, and several cloud providers. These workflows can be executed via Form Buttons, Conditions, Scheduled Jobs, and other workflows.

36. To expedite our lab since we are not focused on Orchestration, several Workflows were built in advance and loaded into your lab instances. Again, since Orchestration is not enabled in your instances, these example workflows mimic Orchestration activities.

37. On the right side, click the **Workflows** tab and enter **Lab 9** in the search box. Left-click on **Lab 9 - Linux Diagnostics Logs** and drag it over the line between Run Script and End until the line turns blue. Release the left button.

38. Enter the following information:

    Workflow: **Leave Alone – Lab 9 - Linux Diagnostics Logs**
    Map return value to: **workflow.scratchpad.confirmMessage**

39. Click **Submit**.

40. On the right side, left-click on **Lab 9 - Linux Restart Server** and drag it below/beside the Lab 9 - Linux Diagnostics Logs activity.

41. Enter the following information:

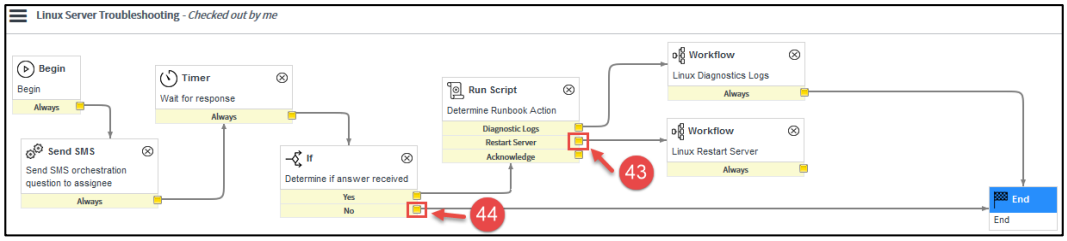
    Workflow: **Leave Alone – Lab 9 - Linux Restart Server**
    Map return value to: **workflow.scratchpad.confirmMessage**

42. Click **Submit**.

43. Left-click the **yellow square** beside **Restart Server** and attach it to the **Linux Restart Server** activity to connect the Restart Server condition to the Linux Restart Server workflow activity.

44. Left-click the **yellow square** beside **No** and attach it to the **End** activity to connect the No condition on the Determine if answer received activity to the End activity.

45. Your workflow should look similar to the example.

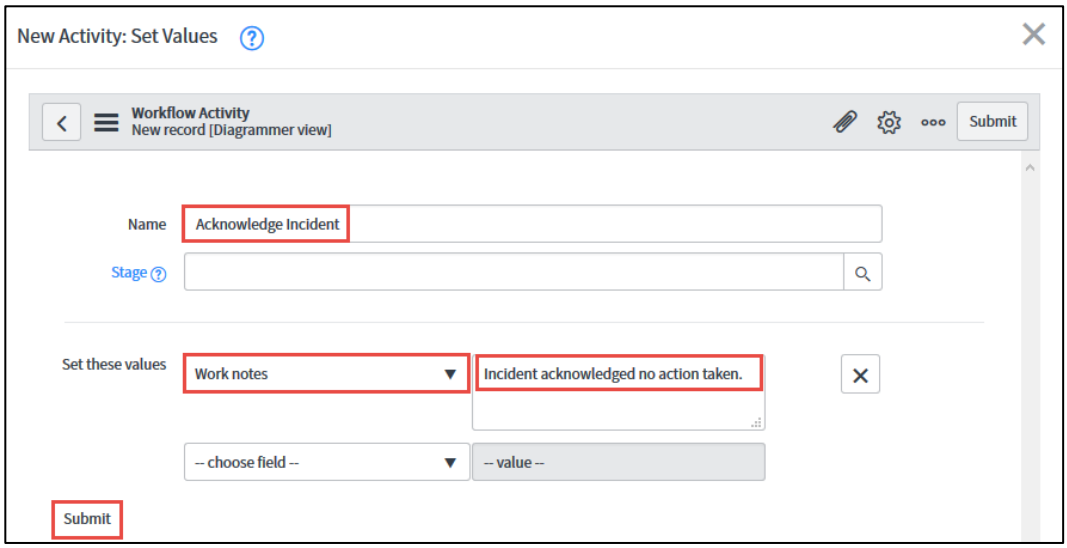


46. Log an update if the incident assignee **Acknowledges** the incident.

47. On the right side of the screen, click the **Core** tab. Under Core Activities\Utilities, left-click on **Set Values** and drag it below the Linux Diagnostics Logs activity.

48. Enter the following information:

    Name: **Acknowledge Incident**
    Set these values: Choose **Work notes** and set the value to **Incident acknowledged no action taken**.



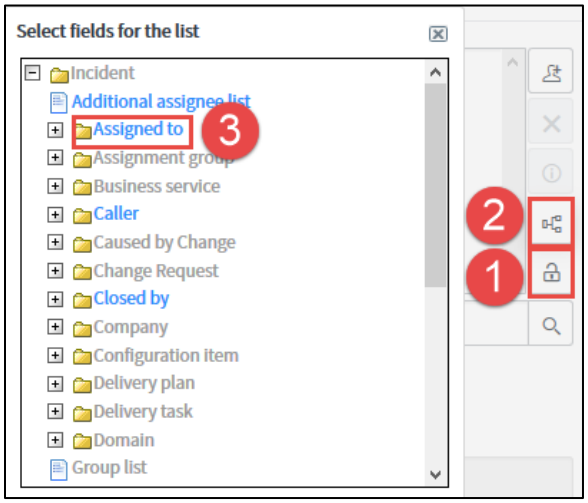
49. Click **Submit**.

50. Left-click the **yellow square** beside **Acknowledge** and attach it to the **Acknowledge Incident** activity to connect the Acknowledge condition to the Acknowledge Incident workflow activity.

51. Under Core Activities\Notify, left-click on **Send SMS** and drag it over the line between Linux Diagnostics Logs and End until the line turns blue. Release the left button.

52. Enter the following information:

    Name: **Send SMS orchestration question to assignee**
    From: **Click the Magnifying Glass and choose your Notify Number**
    To: **Click the Lock button and then the Select Fields button and choose Assigned To**



    Message: Send return value from sub workflows. With scratchpad variables, you do not need to use full name.

```
${confirmMessage}
```

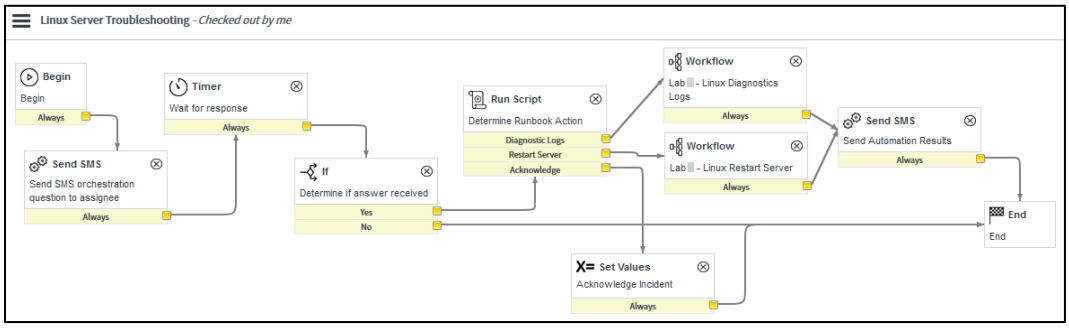
53. Click **Submit**.

54. Connect the **Linux Restart Server** activity to the **Send SMS** activity**.**

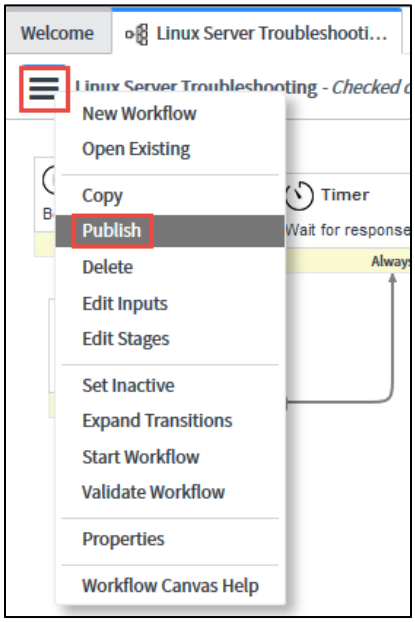
55. Connect the **Set Values** activity **t**o the **End** activity.

56. In this simple example, the workflow ends if a response is not received within 5 minutes. This can be changed to meet your business requirements. Connect the **If No** condition to the **End** activity.

57. Your workflow should look like the example.



58. Click **Workflow Actions** and select **Publish**.



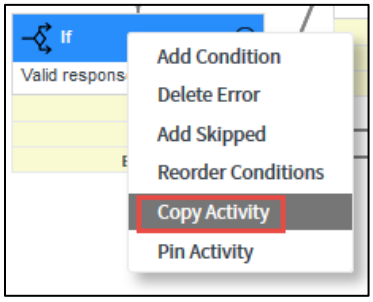
59. You should not receive any validation errors. If you do, you may not have connected all of the activities to one another.

60. Inbound SMS messages from Twilio are processed via a workflow defined in the Notify Number Groups. Your instance is using the **On-Call: Check Assignment Response** workflow. Out-of-the-box, it validates incoming SMS messages expecting only ACC and REJ. In order to make the new codes work, update the workflow to accept the codes.

61. On the right side, click the Workflows tab and enter **Call** in the search box. Click **On-Call: Check Assignment Response**. This opens the workflow in a new tab.

62. Click **Workflow Actions** and select **Checkout.**

63. Right-click on the top portion of the **Valid Response If** activity and select **Copy Activity**.



64. Move the copy down below the existing If activity.

65. Double-click on it to edit it.

66. Change the name of this activity to **Valid Troubleshooting Response?**

67. In the Script, edit line 4 to include the new codes. This line should read:

```
var pattern = new SNC.Regex('/^([0-9]*)(LOGS|RES|ACK)$/i');
```

Note: The script can also be copied from the Lab Management application under the Lab Exercises section by clicking **Lab 9 Orchestration with Notify Scripts**.

68. Click **Update**.

69. Click the line coming from the **No** condition on the original **Valid** response activity so it turns blue. Press the **Delete** button on your keyboard to remove it.

70. Connect the **No** condition to the new **Valid Troubleshooting Response** activity.

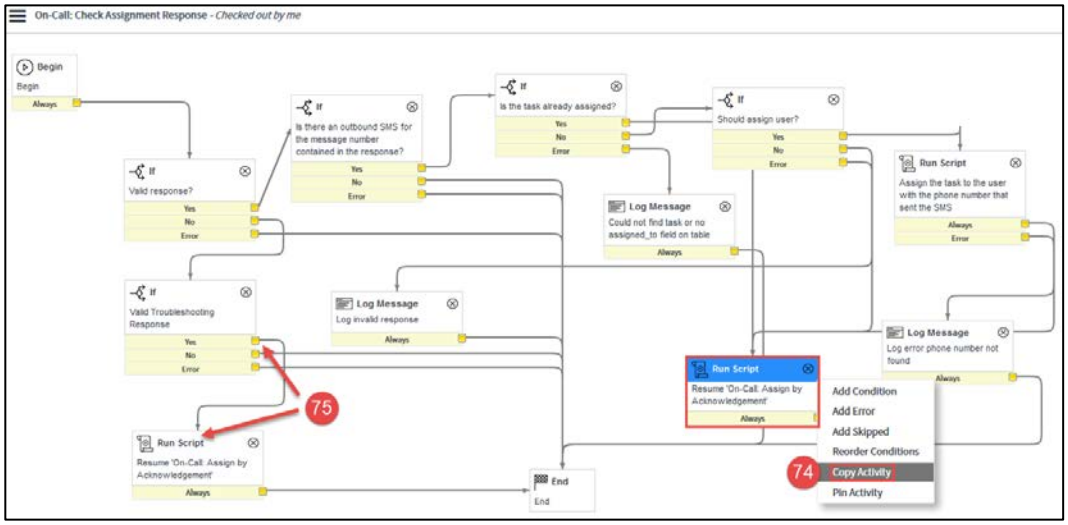
71. This makes the workflow perform one additional check on the SMS response to see if it matches the server troubleshooting codes.

72. Connect the **No** and **Error** on the new **Valid Troubleshooting Response** activity to the **End** Activity.

73. This workflow has a feature that resumes the On-Call workflow after the response is received, which means the other workflow does not have to wait for the timer to complete. You need this same feature for the Linux Server Troubleshooting workflow.

74. Right-click on the top portion of the Run Script Resume '**On-Call: Assign by Acknowledgement**' activity and select **Copy Activity**. Then move this new activity under your Valid Troubleshooting Response activity.

    Note: Your workflow will look slightly different than the one below because of the automation added for Lab 1.1 that set your phone number in a few demo user records.



75. Connect the Valid Troubleshooting Response activity **Yes** condition to the new Run Script you just moved.

76. Connect the new Run Script you just moved to the End activity.

77. Double-click the new Run Script Resume '**On-Call: Assign by Acknowledgement**' to edit it.

78. Set the following values:

    Name: **Resume 'Linux Server Troubleshooting'**
    Script:

```
var message = new GlideRecord("notify_message");
message.addQuery("direction", "outbound");
message.addQuery("phone_number", current.phone_number);
message.orderByDesc('sys_created_on');
message.query();
if (message.next()) {
   var context = new GlideRecord("wf_context");
   context.addQuery("id", message.source);
   context.addQuery("workflow.name", "Linux Server Troubleshooting");
   context.addQuery("state", "executing");
   context.query();
   while (context.next()) {
         var wfScript = new SNC.WorkflowScriptAPI();
         wfScript.broadcastEvent(context.getUniqueValue(), "resume");
   }
}
```

    Note: The script can also be copied from the Lab Management application under the Lab Exercises section by clicking **Lab 9 Orchestration with Notify Scripts**.

79. Click **Update**.

80. Click **Workflow Actions** and select **Publish**.

81. In the real world, there are better alternatives for what we just did, but for simplicity, we used an easy approach.

82. Now test it!

83. In the Filter navigator box, type **incident**.

84. Click **Create New** under the Incident application.

85. Enter the following information:

    Category: **Hardware**
    Configuration Item: **lnux100**
    Short description: **Linux Server Failure**

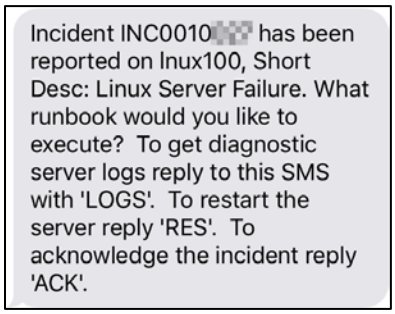
86. Click **Submit**.

87. Automatically the On-Call trigger rule should set the assignment group to **Hardware**. Shortly you should receive an SMS asking you to accept the incident.

88. **Reply** back to the SMS with **ACC**.

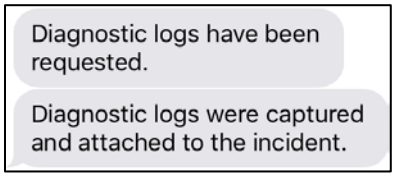
89. The incident is automatically assigned to Beth Anglin.

90. You should receive an SMS message momentarily, asking what runbook action you want to take on the incident.



91. **Reply** to the SMS with **LOGS**.

92. You should receive two SMS messages momentarily, one confirming that you requested diagnostic logs and another that the logs have been captured and attached to the incident.



93. Once you receive the SMS that the Diagnostic logs were captured and attached to the incident message, return to the incident and click **Additional Actions, Reload form**.

94. After the incident refreshes, you should see an attachment at the top and activity messages indicating that the file was attached.

95. Feel free to repeat **Steps 83-88**, choosing a different runbook.

Lab Success Verification

1. Verify that an incident with a Linux Server configuration item that is assigned to someone launches a workflow that allows the assignee to respond and take an automated action to resolve the incident.