Sending SMS through Process Builder

Contents

Sending SMS Through Process Builder Basics

Salesforce Process Builder is a no-coding method to easily handle triggering Outbound Text Messages as well as to process Incoming Messages based on Keywords or other factors. One can literally trigger on any object. Common objects to trigger off of are:

Lead/Contact – Common use cases are when various fields change and you want to trigger an Outbound SMS.

Custom Objects – Similar to Lead/Contact use cases. 360 SMS supports triggered messages from any custom object and its SMS Templates support all custom objects.

SMS_History – Especially useful for incoming SMS – read the message and do something else based on the Incoming Message, either updating the Salesforce record or sending out some other question based on the reply. Useful for Surveys, i.e. Reply with INTERESTED or NO and then SMS_History.Message = INTERESTED updates a field or status in the corresponding Salesforce record.

There are two primary methods of triggering an outbound SMS:

Method #1 – Simple: This is good for customers new to process builder

Method #2 – Apex Class: This is the preferred method as the formula field it uses allows for commented code and you can easily copy/paste it to other process builders. Additionally, the 3rd parameter can accept a TemplateId, the first QuestionId of a Survey or a simple text string of the outbound message if you don't want to use a template or question.

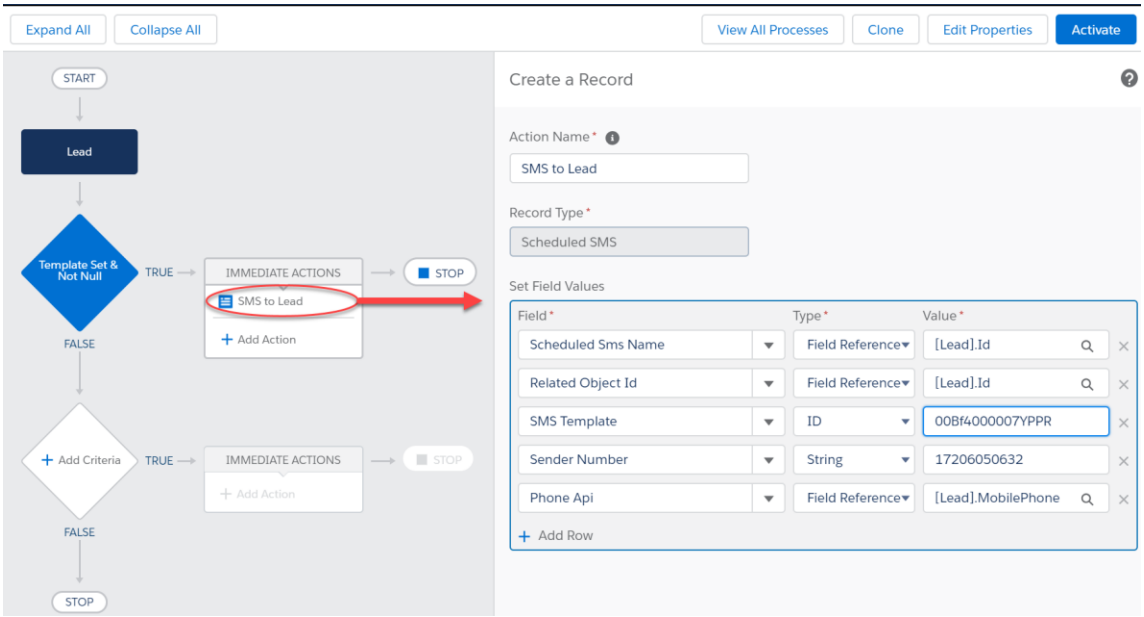


Figure 1 –Method #1 showing the 5 fields required for sending an SMS – note you can also set the QUESTION field which is the first question of the 360 SMS Survey object which will then trigger an automatic Question/Answer survey.

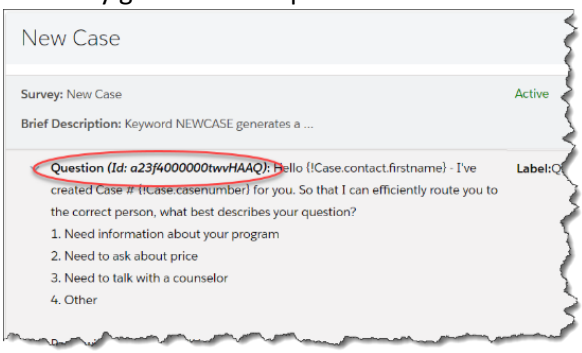
Method #1 – Simple

A few simple settings is all it takes to trigger a message:

1. Set CREATE OBJECT to **Scheduled SMS**
2. **Scheduled SMS Name**: Must be the ID field of the triggered object/record. This is used primarily in conjunction with the SMS Template and must match by object in order for the merging to occur, e.g. Supply a Contact.Id and a matching SMS Template based on the Contact object.
3. **Related Object Id**: Set the Related Object Id to Lead.Id or Contact.Id. Hint you can also set it to other objects to gain visibility to the text conversations, i.e. set it to a Account ID and the SMS History attaches there. However, it will then not set the SMS_History.Contact_Id, so we recommend instead using a Process Builder to attach SMS_History to parent objects.
4. **SMS Template**: Set the ID of the Template to be used. This can be obtained from the URL of the template.
    a. You may also use a reference field such as Contact.SMS_Template (if you've created a SMS Template Id on your Contact – see the section -

Create a Master Send SMS Handler).

5. **Question**: As an alternative to setting an SMS Template, you may set the **Question** field. The QuestionId is the first question of a 360 SMS Survey. This then triggers the first question of the survey and responses are automatically triggered from there on. See the **Surveys** section for more on surveys. We highly recommend utilizing the Survey object because most outbound SMS will likely generate a response which can now be automatically handled.



6. **Phone Api**: Supply the phone field to send the message to. Normally you pull this from the record, i.e. Lead.MobileNumber but it can also be pulled from the Incoming SMS, set with a formula, or for SMS to employees you might use the User.MobileNumber

7. **Sender Number**: Set the Sender Number (this is the number that you are sending FROM) this can also be a referenced field such as Lead.Owner.Phone (so as to send from different sales people) or a formula such as different geographies using different numbers. If you only have one outbound number in your org, the field is optional and need not be supplied.

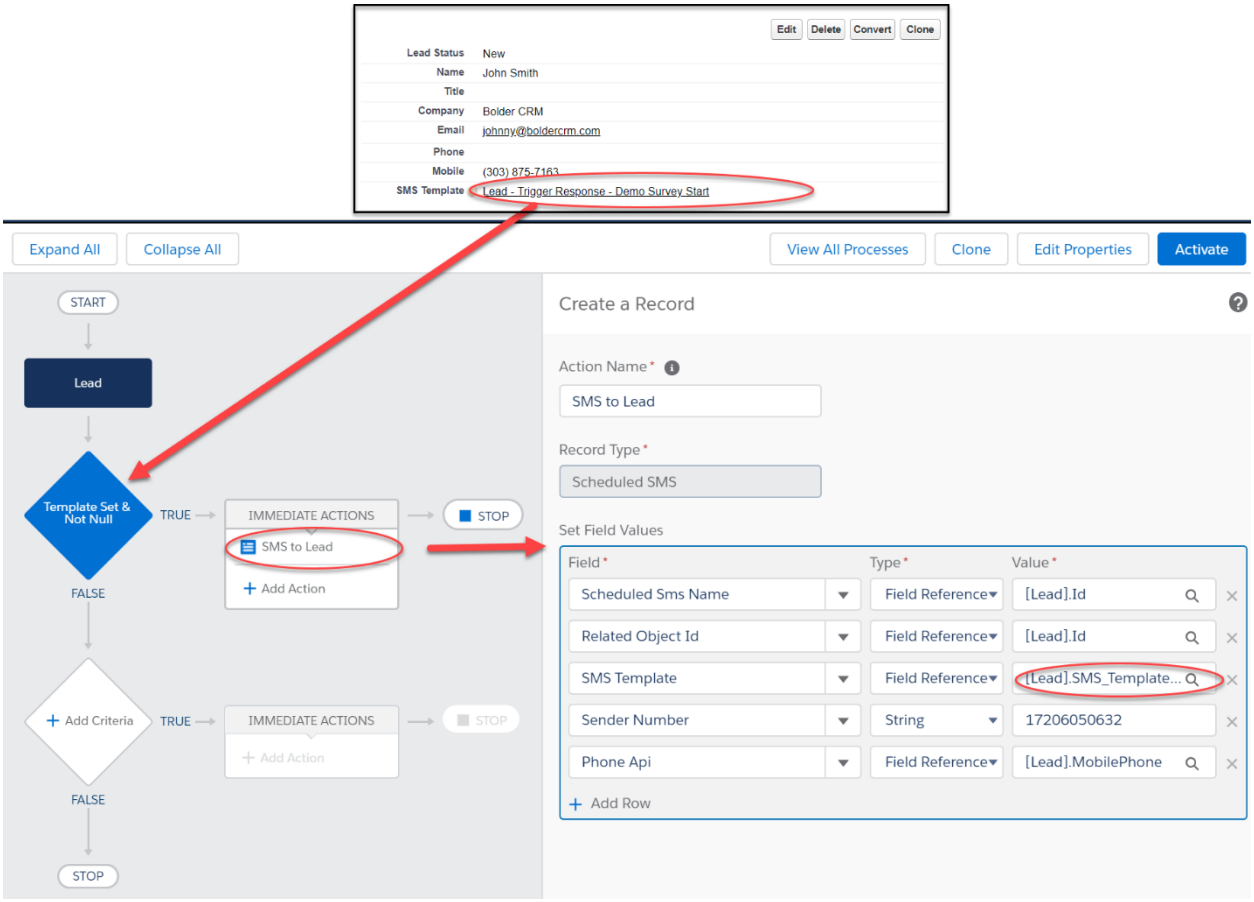


Figure 2 - In this scenario we are pulling the Template from a custom field we placed on the Lead for even easier automation. Now you can have other process builders that only need to set the Lead.SMS_Template and that alone will trigger an outbound SMS.

Relating Outbound SMS to an Alternate Object

The **Related Object Id**, which is available only for Method #1, has some interesting uses, primarily when it is desired to link the message to an alternate object than where it is being initiated from and where its template is based on.

1. Linking a message to the parent ACCOUNT object of a CONTACT instead of the contact

2. Linking a message initiated from an Opportunity to a primary Contact or Account

3. Or the big use case is when triggering messages to internal users in which case you should use the **Related Object** to link the message to the USER object rather than the main object so that the internal notification does not appear in the Contact/Lead SMS History and look like a message was sent to the customer.

Below is an example where we are triggering a text message when a new Lead is created from the Web Site. Note how the Phone API is pulled from **Lead.Owner.User.MobilePhone** and the Related Object Id is set to **Lead.OwnerId**. This allows us to use a template based on the Lead Object so we give the Lead.Owner the lead details but it keeps the SMS History from showing up under the Lead itself.

Note that the SMS Template's object must match the object of the ID supplied in the Scheduled SMS Name field, it has nothing to do with the Related Object Id.

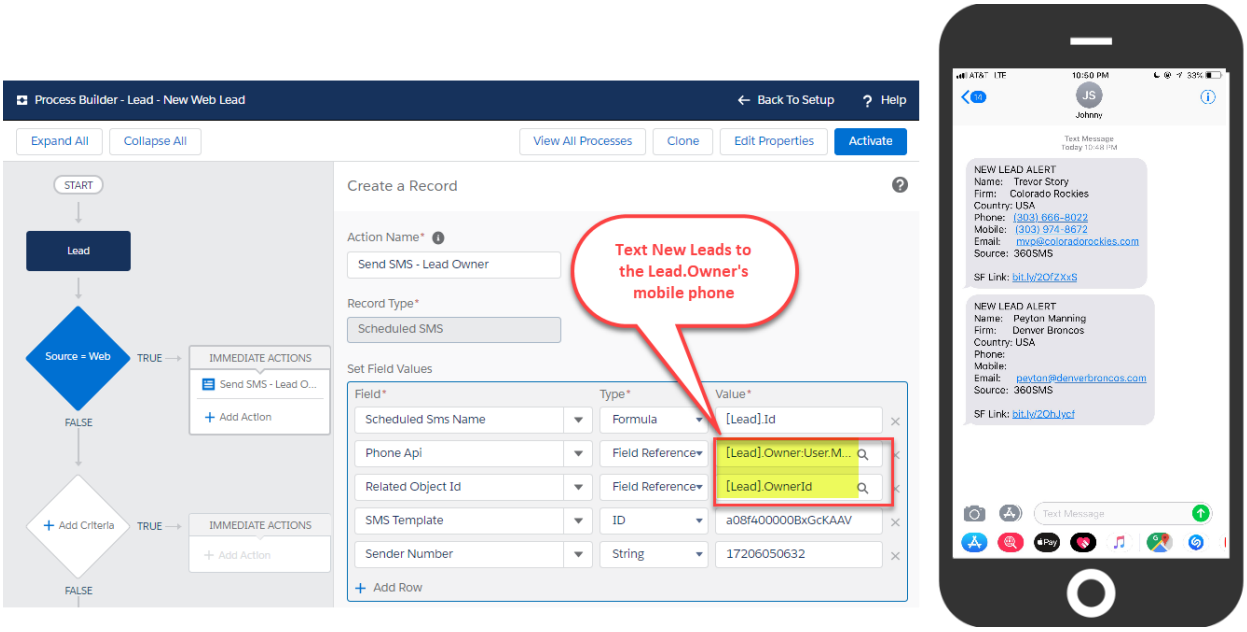


Figure 3 - Typical internal alert via SMS sending to an employee but triggered from the Lead creation

Method #2 – Apex Class

360 SMS also has an Apex class that can either be called in Process Builder, Flows or in trigger code. There is an Apex class for sending regular **SMS** and another one for sending **MMS** which includes a parameter for the picture or file.

The Apex classes accept parameters in a comma separated string that you pass to the **param** field of the Apex class. This method is nice for copying/pasting into sophisticated workflows. One of the best things about this method is that Salesforce allows comments in formula fields, so we strongly recommend commenting your formulas using the **/* some comment */** syntax.

For the regular **Send SMS From Process Builder** Apex Class the string of parameters is defined as:

Param1: Id of the primary object you are triggering from – this must match your Template object and it will be the primary object that the outbound SMS will relate to.

Param2: The API name of the phone field for that object or it can be any value evaluating to a valid phone number

Param3: This parameter can take an SMS TemplateId, a QuestionId or a String.

- **TemplateID:** Use a Template Id pulled from the URL of an SMS Template – its object must match the object defined in param1.
- **QuestionId:** Use a Question Id which is typically the first Question of a Survey. This will trigger survey question #1 and all responses will be handled automatically.
- **String:** You may pass a string in this parameter if you do not want to use templates, this works well for simple SMS messages. The string also supports merge tags. e.g. 'Okay {Contact.firstname} – this is a msg w/o a template'

Param4: Outbound phone number, if blank it sends the default phone number for the org or the first phone number found in the 360 SMS User Configuration tables for the current user.

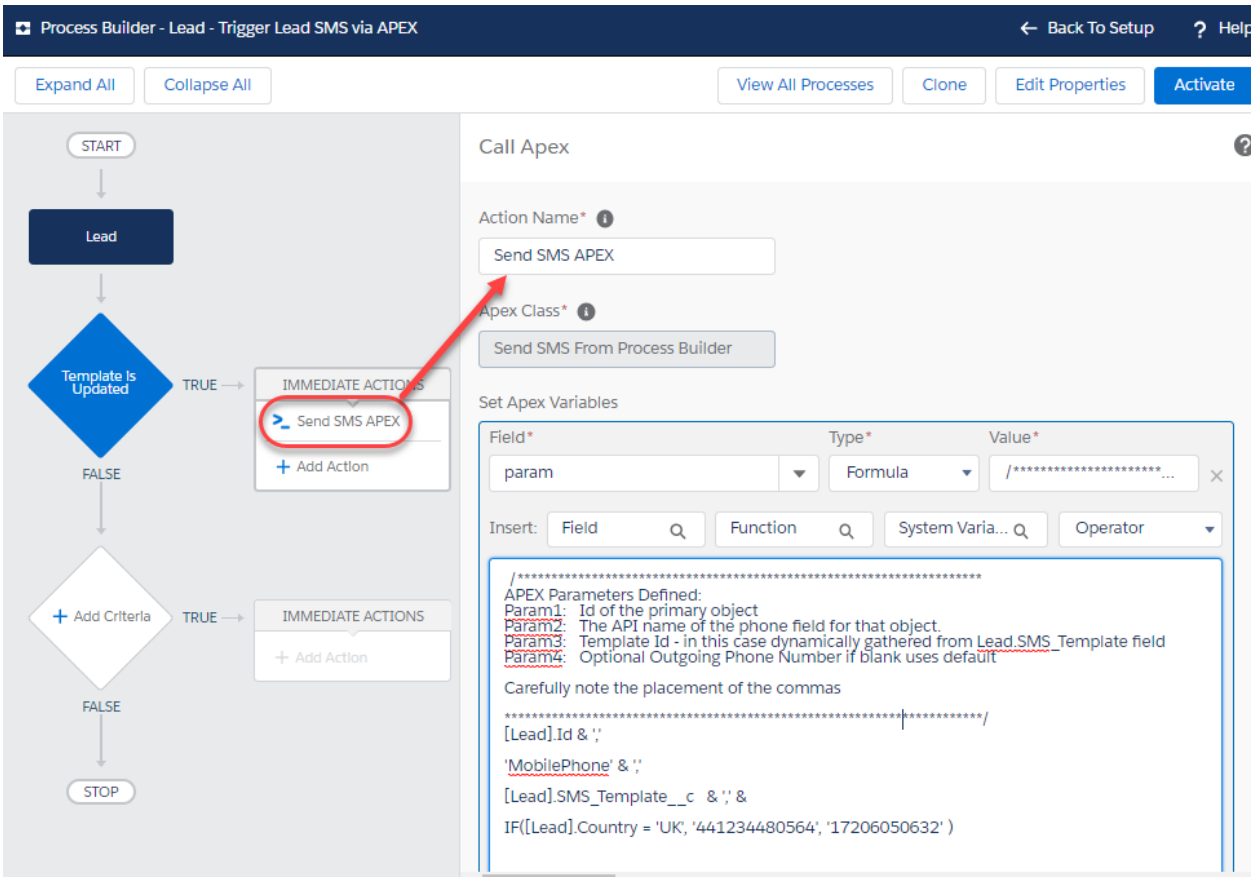


Figure 4 - Code example of sending regular SMS via the "Send SMS From Process Builder" Apex class

Below are a couple of code snippets for easy copy/pasting

```
/*********************************************************************
APEX Parameters Defined:
Param1:  Id of the primary object

Param2:  The API name of the phone field for that object or an actual phone value

Param3:  Can be one of three:
      1. Template Id - hardcoded or referenced like Lead.SMS_Template
      2. Question Id - 1st question of a Survey  - a23f4000000uObmAAE
      3. Straight Text - can include merge tags

      This example is calling the 1st question of a Survey

Param4:   Optional Outgoing Phone Number if blank uses default. Here we have a
different # for UK customers than USA

Carefully note the placement of the commas

*********************************************************************/
[Lead].Id & ',' &
'MobilePhone'  & ',' &
'a23f4000000uObmAAE' & ',' &
IF([Lead].Country = 'UK', '441234480564', '17206050632' )
```

```
/*********************************************************************
APEX Parameters Defined:
Param1:  Id of the primary object

Param2:  The API name of the phone field for that object or an actual phone value

Param3:  Can be one of three:
      1. Template Id - hardcoded or referenced like Lead.SMS_Template
      2. Question Id - 1st question of a Survey  - a23f4000000uObmAAE
      3. Straight Text - can include merge tags

      This example is referencing a custom field on the Lead which is a lookup to the
      actual SMS Template Id. That's a nice trick so you can have other PB's simply
      setting this field and triggering this code to do the actual sending SMS

Param4:  Outbound Number

Carefully note the placement of the commas

*********************************************************************/
[Lead].Id & ',' &
'MobilePhone'  & ',' &
[Lead].SMS_Template__c  & ',' &
'17206050632'
```

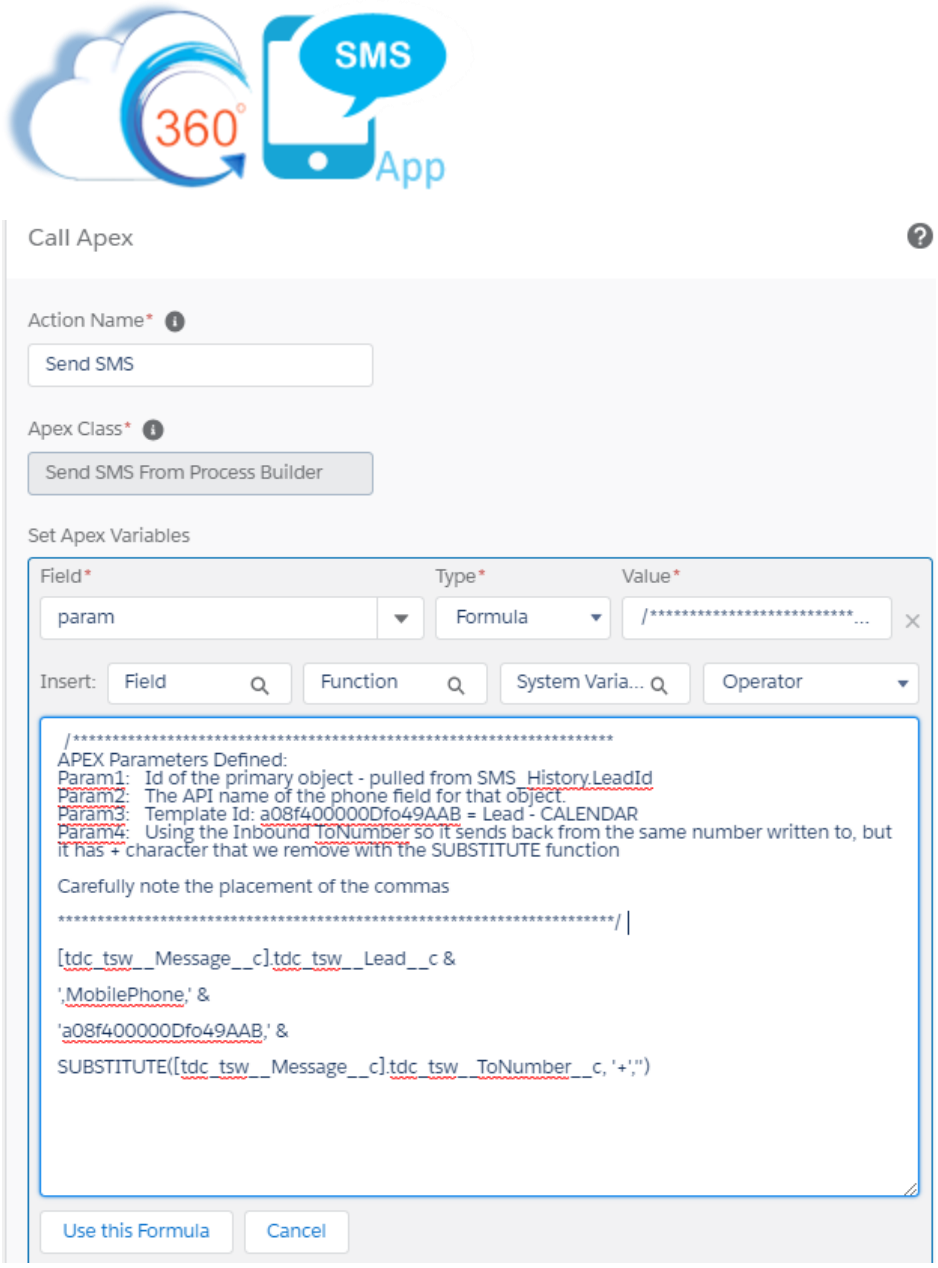


Figure 5 - Example APEX when triggering a reply from an Incoming SMS History

Send MMS via Process Builder

MMS is the term for sending or receiving PICTURES in a text message. 360SMS provides a separate APEX class named **Send MMS From Process Builder** with an extra parameter for the Saleforce Document Id of the picture to send. The string of parameters are described below and shown in *Figure 7*.

Param1: Id of the primary object you are triggering from – this must match your Template object and it will be the primary object that the outbound SMS will relate to.

Param2: The API name of the phone field for that object.

Param3: This parameter can take an SMS TemplateId, a QuestionId or a String.

TemplateID: Use a Template Id pulled from the URL of an SMS Template – its object must match the object defined in param1.

QuestionId: Use a Question Id which is typically the first Question of a Survey. This will trigger survey question #1 and all responses will be handled automatically.

String: You may pass a string in this parameter if you do not want to use templates, this works well for simple SMS messages. The string also supports merge tags. e.g. 'Okay {Contact.firstname} – this is a msg w/o a template'

Param4: Optional Document ID of the picture or file to send

Param5: Optional originating phone number, if blank it sends the default phone number for the org or the first phone number found in the 360 SMS User Configuration tables for the current user.

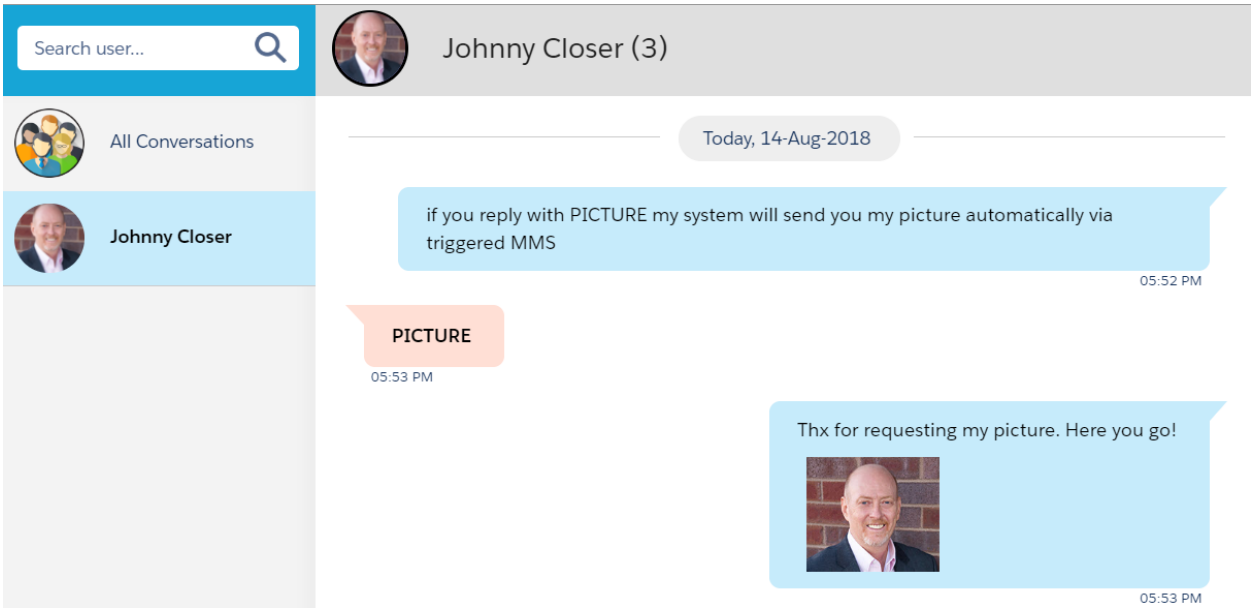


Figure 6 - MMS automation with keyword "Picture" sending pic of the Lead.Owner

Figure 7 also demonstrates a completely dynamic solution where the picture is derived by navigating to a custom field on the User record via SMS_History.Owner and the outbound phone number is also gathered from the User record. Most of the time you will be dynamically setting the Pictures and Outbound Number.

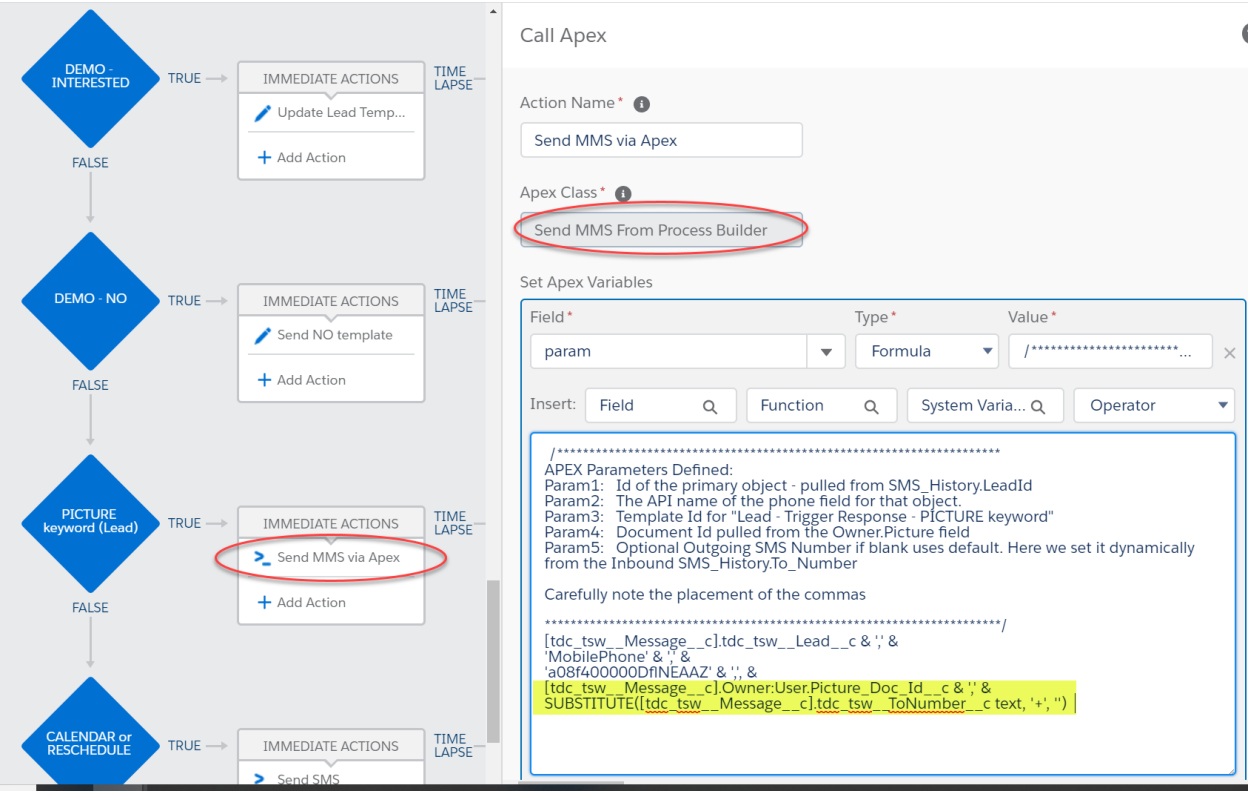


Figure 7 - PB code for executing triggered MMS via the "Send MMS from Process Builder" Apex class – this example is triggering off an inbound message with keyword "PICTURE". We obtain the picture by traversing up to the USER object and accessing a custom Picture_Doc_Id field. We use the SMS_History.To_Number to send back the reply.

Dynamically setting the Outbound Number

No matter which method one uses, it is common that the automation should send the Outbound SMS from the record owners unique SMS Number. Many orgs use separate numbers for each user. These are defined in the SMS Setup → User Configuration which is not accessible via process builder. However, with a simple customization to your USER object you can make your Process Builders dynamically obtain the Outbound SMS Number parameter that either method makes available as an optional parameter.

Simply, create a custom text field named something like User.**SMS_Number**. Then copy the number associated to each user into the field. DO NOT attempt to use the standard User.MobilePhone field as Salesforce formats this number on you, such as (720)605-0632. The number needs to be completely unformatted and have the country code prefix, i.e. 17206050632.

As shown in *Figure 7* you can now traverse to the User table via the SMS History.Owner or Lead.Owner/Contact.Owner and get the number from your custom field.

A second common scenario is to dynamically set the SMS Number parameter based on the Incoming Message. This is common when responding to Keywords. In this case, you don't need to lookup the number from a user table, you simply need to get it from the SMS_History.**To_Number** field (the number that the customer wrote to). However, be careful as the value will have a "+" character in front of it which is invalid for an outbound number, so you must use the **SUBSTITUTE** function as shown below to remove the +. The formula is provided below for easy copy/pasting.

```
/*********************************************************************
APEX Parameters Defined:
Param1:   Id of the primary object - pulled from SMS_History.ContactId
Param2:   The API name of the phone field for the contact object.
Param3:   Can be a TemplateId, QuestionId(first Question of a Survey) or straight text if you
          don't want to use a Template or Question
Param4:   Outgoing Phone # - pulled from Inbound SMS_History.To_Number but we have to
          remove the + that is inherent with inbound numbers

Carefully note the placement of the commas
*********************************************************************/

[tdc_tsw__Message__c].tdc_tsw__Contact__c  & ',' &

'MobilePhone' & ',' &

'a08f400000Dfo3fAAB' & ',' &

SUBSTITUTE([tdc_tsw__Message__c].tdc_tsw__ToNumber__c , '+', '')
```

Dynamic MMS – such as sending a picture of a particular user

Like a dynamic outbound number, one can create a custom field on the User record that holds the Document Id of a previously stored picture.

In this example, we simply uploaded a Picture to the Salesforce Document object and manually copy/pasted the actual ID of the picture into a custom field named User.**Picture_Doc_Id**. We obtained the Document Id from the URL when we opened the picture.

For ease of copy/pasting the code for the Dynamic MMS has been provided below.

```
/*********************************************************************
APEX Parameters Defined:
Param1:   Id of the primary object - pulled from SMS_History.ContactId
Param2:   The API name of the phone field for that object.
Param3:   Template Id: a08f400000DflORAAZ = Contact - Event Reminder
Param4:   Document Id pulled from the Owner.Picture_Doc_Id field
Param5:   Optional Outgoing Phone # - pulled from Contact.Owner --> User.SMS_Number (custom)

Carefully note the placement of the commas
*********************************************************************/

[tdc_tsw__Message__c].tdc_tsw__Contact__c   & ',' &
'MobilePhone' & ',' &
'a08f400000DflORAAZ'  & ',' &
[tdc_tsw__Message__c].tdc_tsw__Contact__c.Owner.Picture_Doc_Id__c  & ',' &
[tdc_tsw__Message__c].tdc_tsw__Contact__c.Owner.SMS_Number__c
```

Create a Master Send SMS Handler

Figure 2 above introduced the concept of creating a single Process Builder that triggers via the OnChange of a custom field which you add to your object. We recommend adding the **SMS_Template_Id** lookup field to a Lead, Contact or any custom object. Then as shown in *Figure 2* you can have numerous process builders that trigger outbound messages but all you will need to do is update the Contact.SMS_Template with whatever template you want to send. This centralized approach means that you won't have to create the same Send SMS action whether that be Method #1 or Method #2 over and over again.

Of course, there will be many times when you will not want to use your Master Send SMS Handler such as when the outbound number needs to come from a different number or perhaps when you need to trigger an MMS.

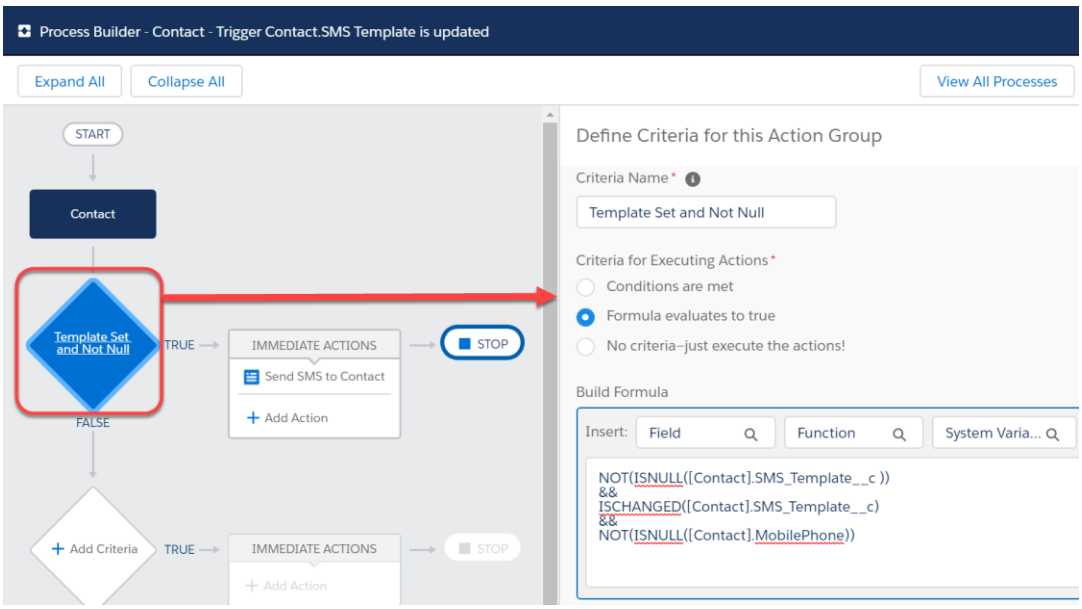


Figure 8 - When the SMS Template is changed - trigger the outbound SMS, Figure 2 shows the Immediate Action

Figure 9 shows a perfect example where we have a survey with multiple answers and it needs to trigger a different template per response. It would be a hardship to write the APEX code for each possible answer over and over again. So instead, we just set the SMS_Template for the contact and let our other Process Builder do the work!

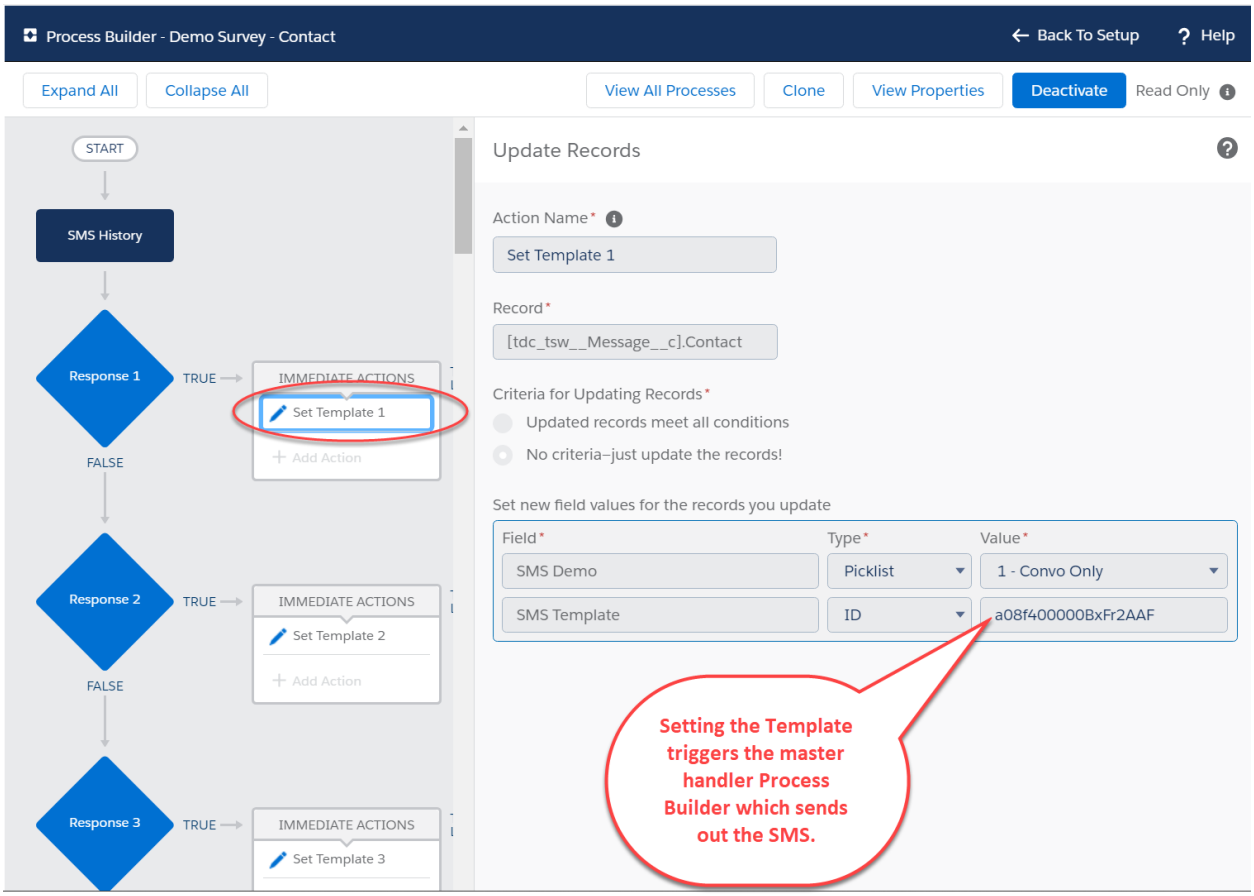


Figure 9 - Demonstration of the easier way of triggering an SMS via a change to the Contact.SMS_Template_Id (custom formula and matching PB)

Surveys

The 360SMS Survey structure is a powerful feature that automates the entire Question/Answer dialog **without** the use of Process Builder or Templates. Answers and their related question are written to the SMS History like everything else in addition to being stored in a **Survey Response** object for even greater reporting and automation potential. Survey's allow a question to have multiple answers which then branch to additional questions with their own multiple answers and as many branched questions/answers as your heart desires.

A survey can be triggered either via traditional Process Builders using the methods described in Method #1 or Method #2 above or a survey can be triggered automatically by defining an inbound keyword.

Survey's work best when the question is presented with clear answer choices such as "Reply YES or NO" or with multiple choice answers where the call-to-action is to "Reply with a number" or "Reply with a letter or combination of letters."

Lastly, survey replies can easily update field values in Salesforce objects using Process Builders on the SMS History object to inspect the incoming answer to a specific question. Read more in the *Updating Salesforce Fields with Survey Answer Responses* section below.

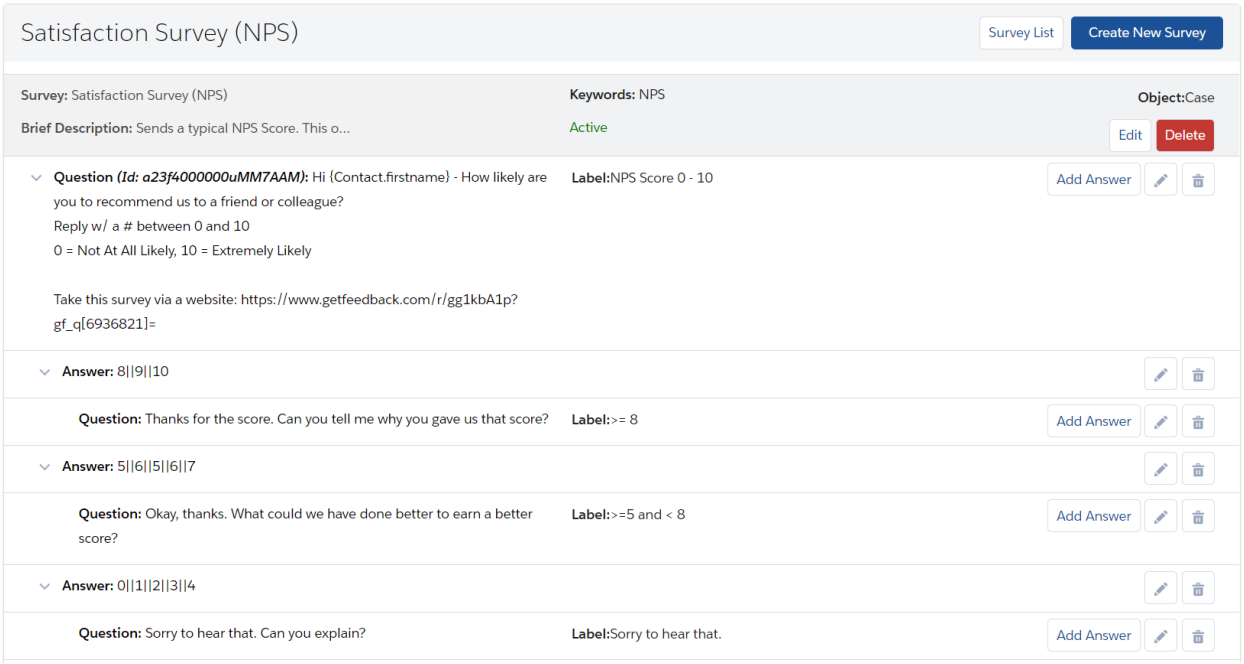


Figure 10 - Typical NPS score survey (Customer Satisfaction Survey)

Survey's also provide a syntax for handling variations of a specific answer, so the system gets smarter over time. Variations of an answer are separated with two pipe symbols ||. As an example, in response to the question "Mail, Email or Text?" (shown in **Figure 11)** we have handled the possible answers to **EMAIL** as:

EMAIL||E-Mail||Email It|| Email me

The responses and your answer definitions are **not** case sensitive

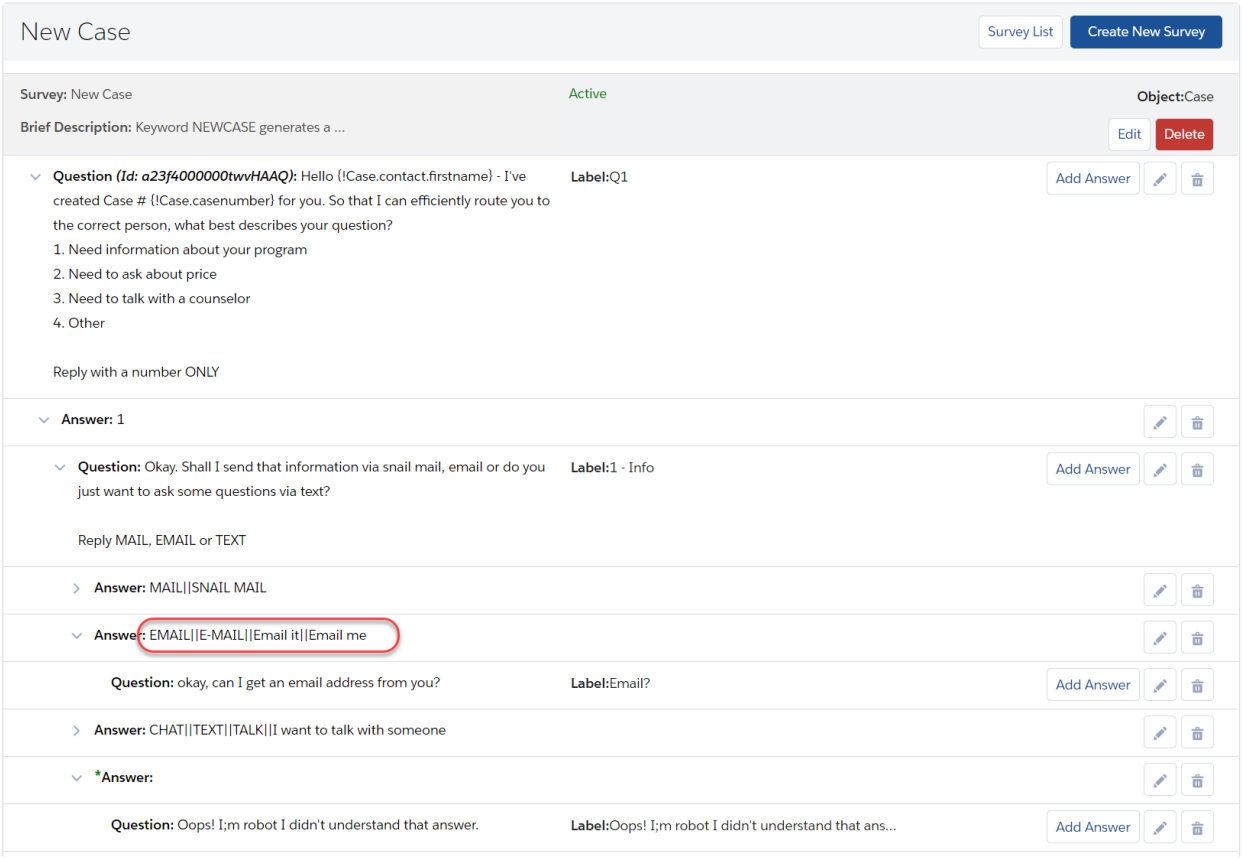
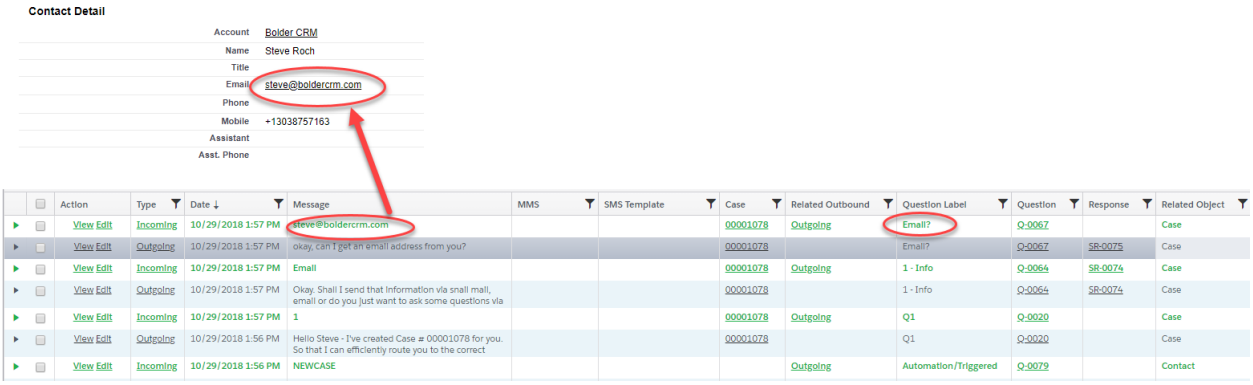


Figure 11 - Sample Survey for when a new CASE is generated. Note the alternate answer syntax

Updating Salesforce Fields with Survey Answer Responses

Commonly one will want to take action with the data gathered via an SMS Survey. The elegant design of the Survey architecture and the SMS History object make this a snap.

Because both the question and the response are written to the SMS History object, there is context to the incoming SMS History, i.e. the Incoming SMS has the Question what was asked as a field on the SMS History record. One need only create a Process Builder or Flow detecting the Incoming SMS to a particular Question and then take action with all the power of Process Builder or Flows such as to Update Record, Create Record, Email Alert, etc..



Below is a code snippet from the process builder receiving an Email Address from the question “What’s your email?”

```
/*****************************************************************************
Incoming message answering the Survey Question - What's your email?
Survey Question:  a23f4000000uGq2AAE = Email?  for the NEWCASE survey
*****************************************************************************/
[tdc_tsw__Message__c].Name = 'Incoming' &&
[tdc_tsw__Message__c].tdc_tsw__Question__c  = 'a23f4000000uGq2AAE'  &&
NOT(ISNULL([tdc_tsw__Message__c].tdc_tsw__Contact__c ))  &&
/*** Make sure it's a good email address **/
CONTAINS([tdc_tsw__Message__c].tdc_tsw__Message_Text_New__c , '@')  &&
CONTAINS([tdc_tsw__Message__c].tdc_tsw__Message_Text_New__c , '.')  &&
NOT(CONTAINS(TRIM([tdc_tsw__Message__c].tdc_tsw__Message_Text_New__c) , ' '))
```

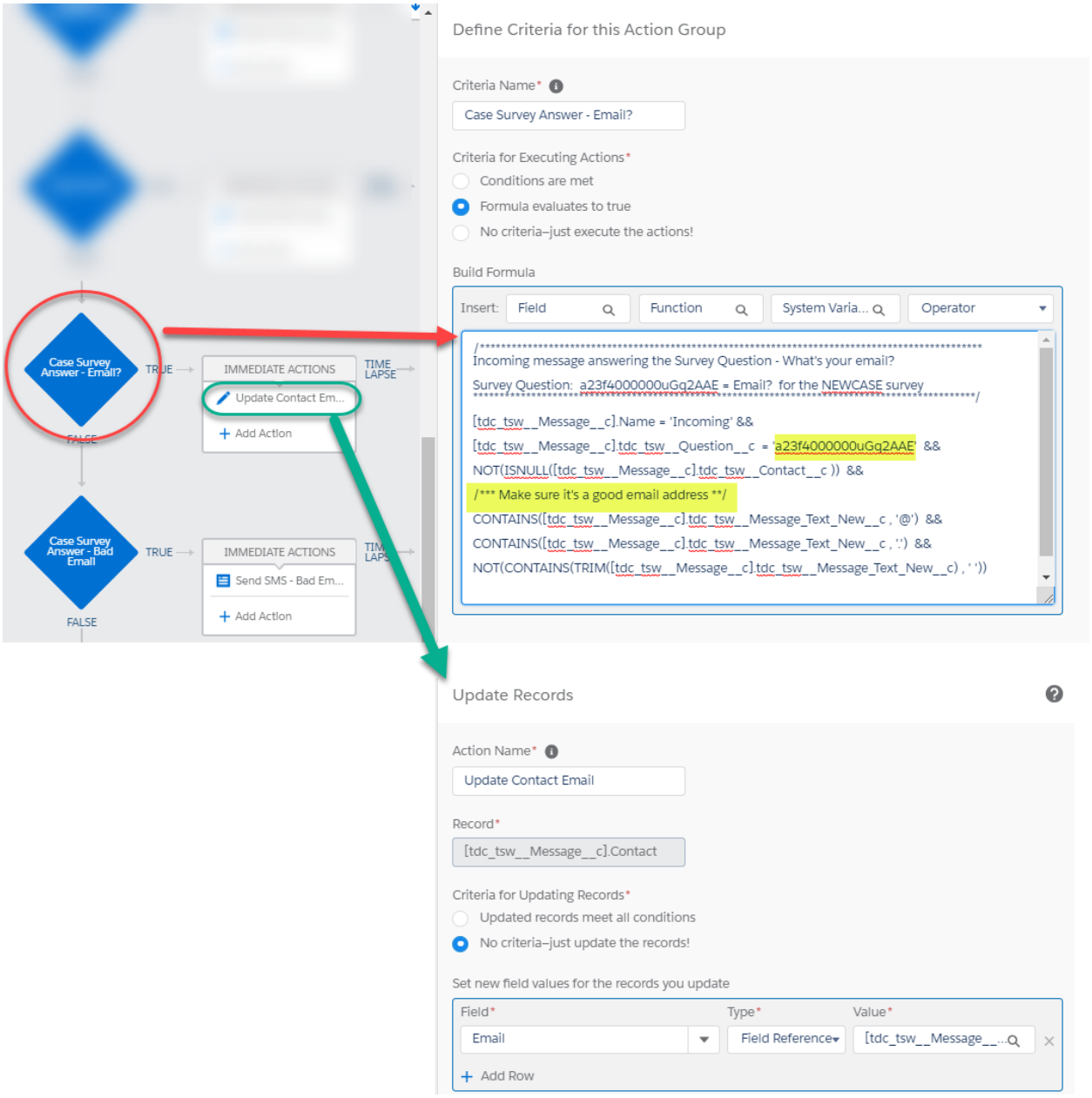


Figure 12 - Process Builder detecting a Survey Answer and updating a Salesforce field

Using Salesforce Flows

The Salesforce Flow technology is worth a short discussion as it provides considerable more power than Process Builder such as the ability to Lookup Records and Mass Update them.

In the examples below we demonstrate receiving an email address via an Incoming SMS in response to a template which says "***I don't recognize your number, can I get your email address so I can look you up by email?***"

In this particular workflow, we have a new unknown number writing into our Salesforce system. In a previous Process Builder we have created a new Lead record for this incoming SMS and we start asking them questions to fill out the record such as Name, Email and Company. However, here we use a Flow called from a process builder to take the email address and perform a Record Lookup against the contact object. If we find a record, we will instead send back a template that says, "***Found you Contact.Name!***" and then proceed with the original keyword that started the whole process.

Figure 13 shows how a normal process builder can call a Flow passing in parameters that we gather from the SMS_History record. *Figure 14* then shows the details of the flow where it:

1. Lookups up the email from the Contacts object
2. If a contact is found it re-links all the SMS_History that got linked to the dummy lead record which was created when the SMS from the unknown record first came in.
3. Finally we can send the outbound SMS reply either via the same APEX methods as a Process Builder uses or in this case we use the technique shown in *Figure 2* where we only update the Contact.SMS_Template field which in turns triggers our Master SEND SMS Handler.

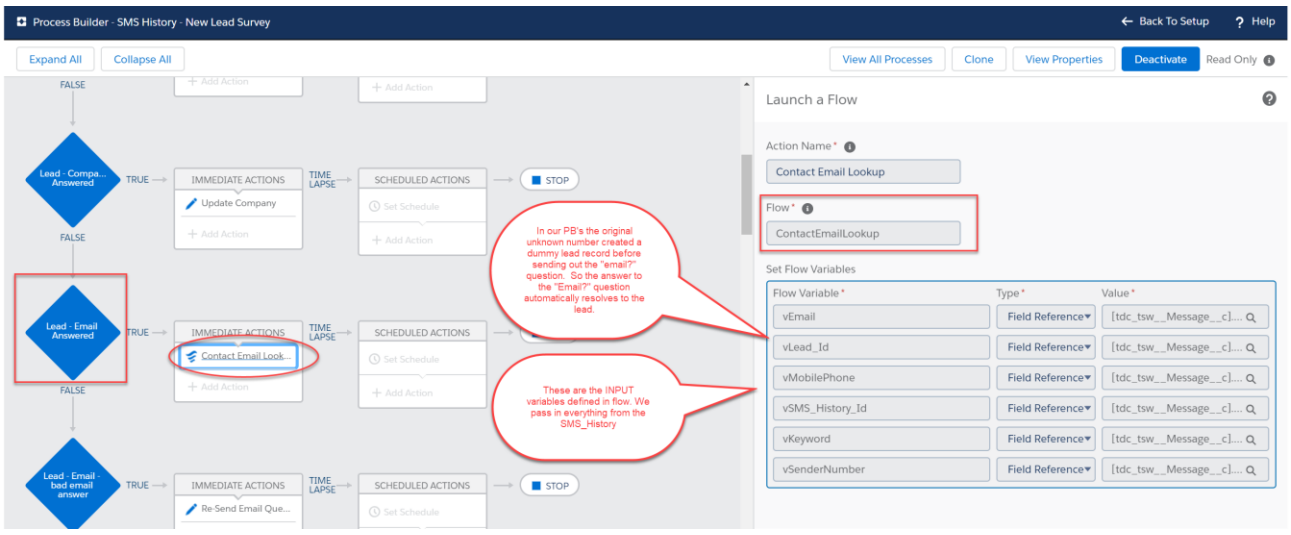


Figure 13 - Example of an SMS_History Incoming process builder triggering a call to a flow.

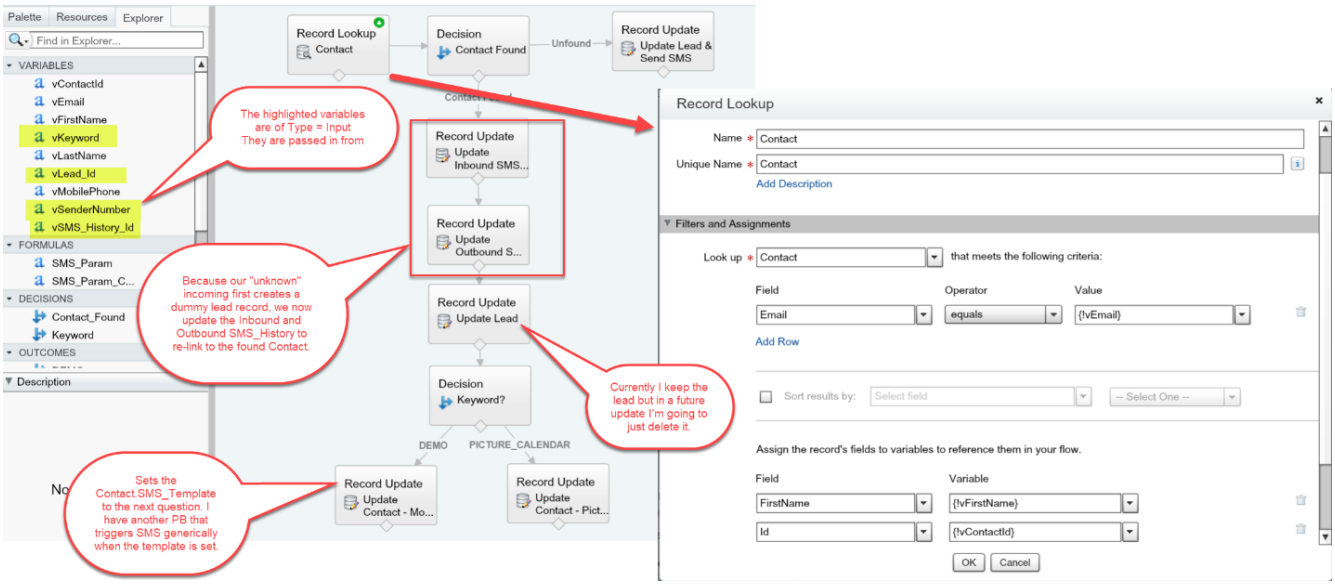


Figure 14 - A flow which does a Record Lookup based on a response from an inbound SMS with an email address

Drip Campaigns

A common automation task is what is commonly referred to as a **Drip Campaign**, whereby you place a Contact/Lead into a Campaign or even trigger any Process Builder and then you want to keep Texting the Contact/Lead periodically until they respond to your Call-To-Action such as replying with a keyword or clicking a trackable link.

There are of course other methods to construct a Drip Campaign, but this method described below is the easiest to maintain.

In this example, we have created two custom fields on the Campaign_Member object and one custom field on the Contact:

Campaign_Member.SMS_Template_Id: Lookup field to the SMS_Template – we move the person through the drip by setting the SMS_Template to a new template at each stage. You could also avoid this step by using the APEX method of sending an SMS at each stage. This method triggers a call to another PB which looks for changes to Campaign_Member.SMS_Template and triggers the SMS, just for easier centralized coding.

Contact.SMS_Stop_Drip: Checkbox field on the Contact record, since incoming SMS will link to the Contact we only have scope to the Contact record and we can mark this field TRUE when they reply to our message via our Incoming Keyword Handler Process Builder.

Campaign_Member.Stop_Drip_Contact: A formula field back to the Contact.SMS_Stop_Drip so that the main PB that handles the Campaign.SMS_Template OnChange checks the field before deciding to send the next drip.

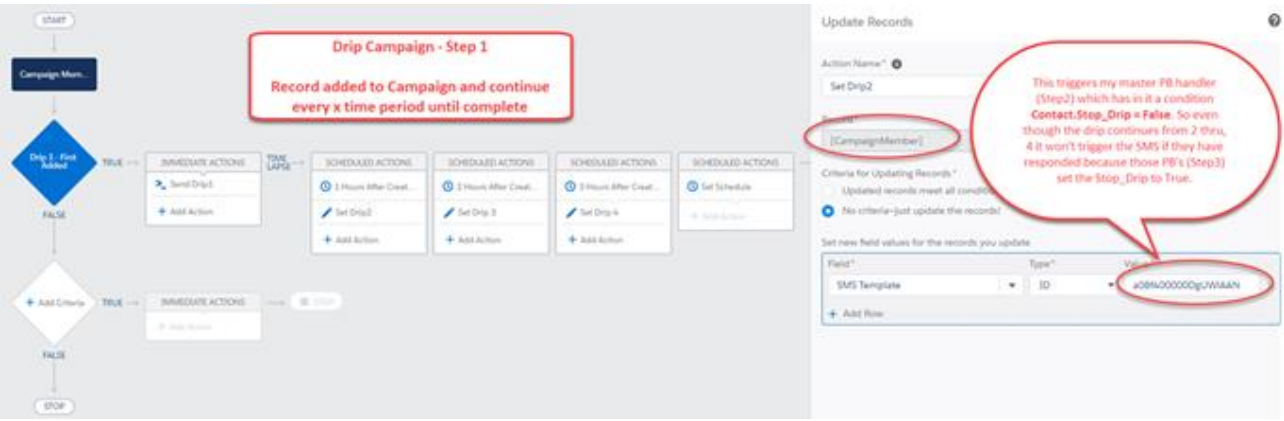


Figure 15 - Step 1: Define the drip campaign - usually starts when a Contact/Lead is added to a Campaign (via CampaignMember)

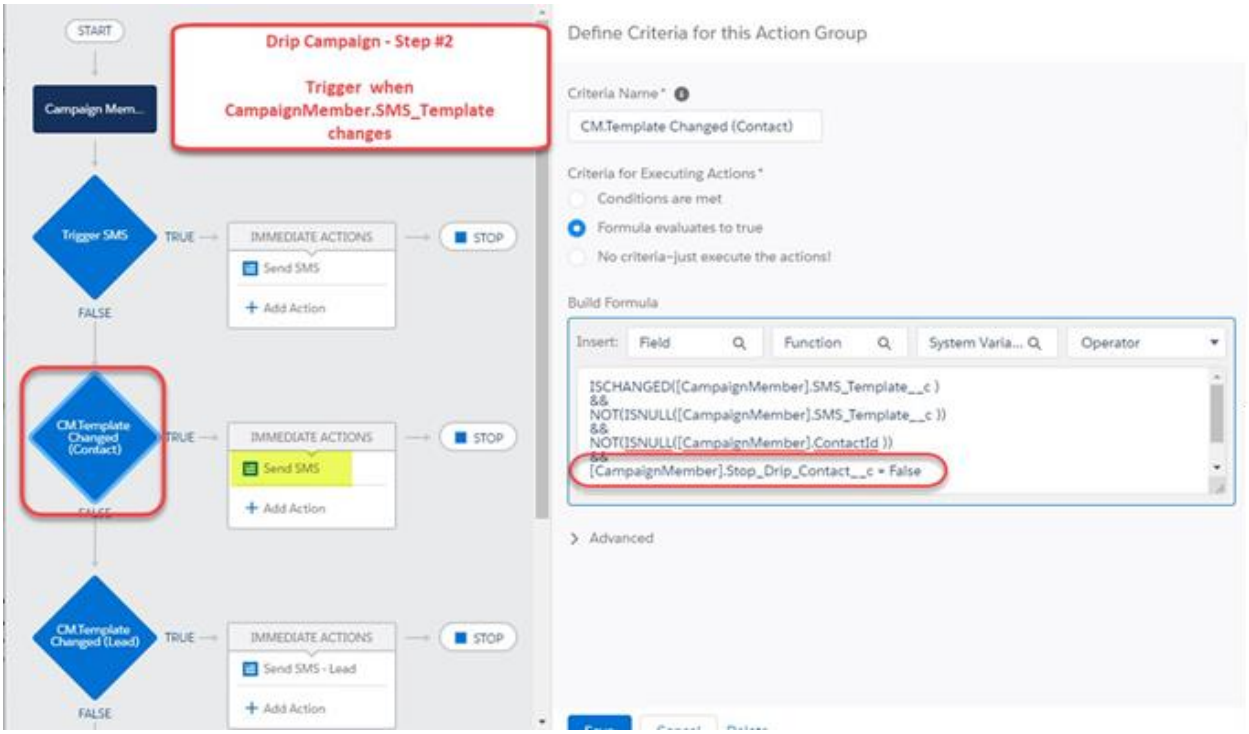


Figure 16 - This is the master Process Builder that handles all changes to the CampaignMember.SMS_Template and triggers an outbound SMS

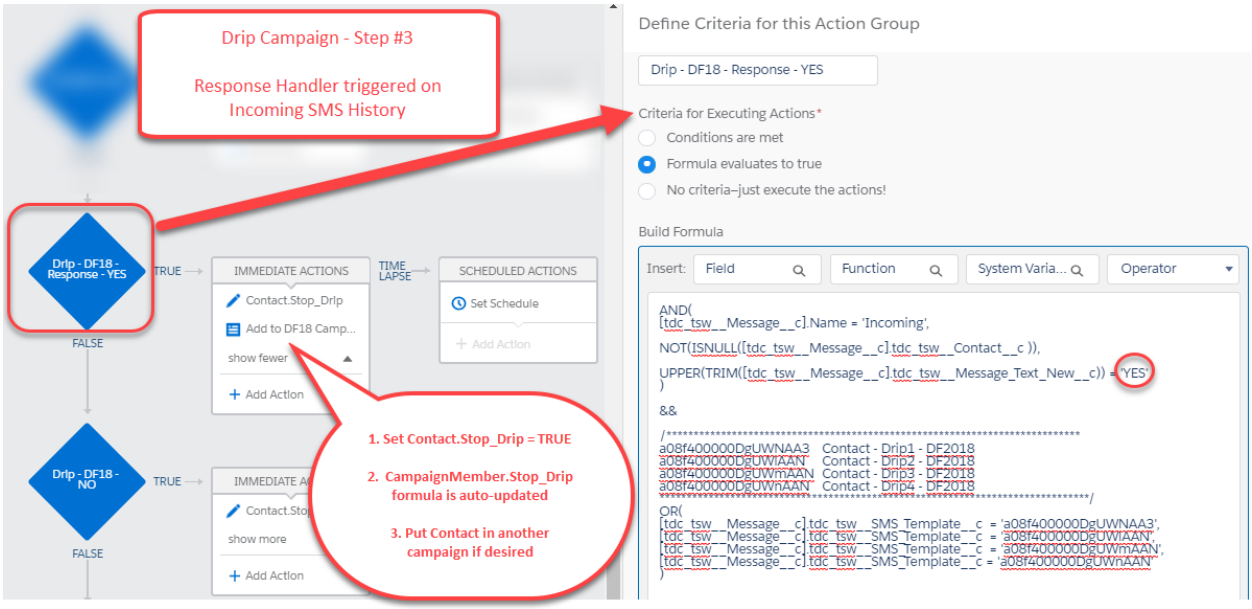


Figure 17 - This is the Response Handler handling the keyword responses that stop the drip

Triggered Texting to Internal Users

Some customers want to use the power of texting to change out their old email alert systems to instead using Text Alerts to the internal users. In the scenario below, we see a use case for a New Lead text alert sent to the Lead.Owner's mobile phone with a hyperlink to the Salesforce record so that he/she can open it quickly in Salesforce1.

Note that we are triggering the message with the Lead.Id so that we may use a Lead based template which merges in details about the lead and most importantly we set the Phone API field to get the value from the Lead.Owners Mobile phone field, i.e. Lead.Owner.User.MobilePhone. Thereby sending the LEAD details to the Owner's personal cell phone.

Important note: Set the Related Object Id to Lead.OwnerId (a user id) so that the alert does not appear in the SMS History for the lead. Otherwise it looks like this is a message you texted to the customer.

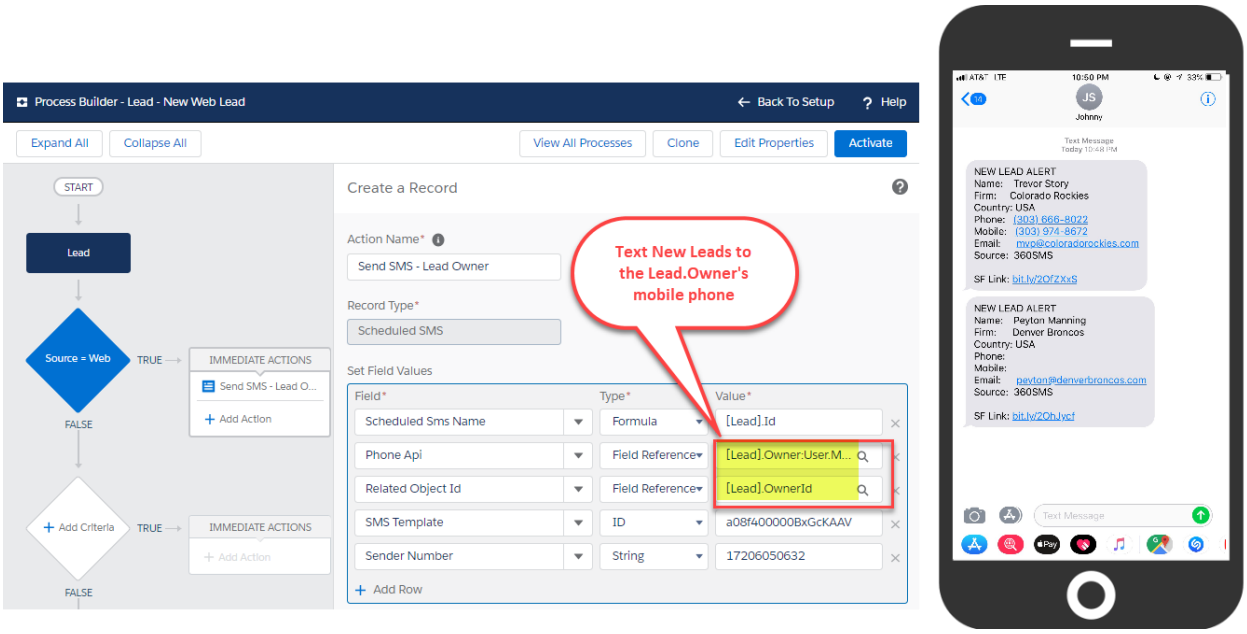


Figure 18 - Sending text alerts to internal users dynamically using the Lead.Owner.User.MobilePhone