Sending SMS through Process Builder

Updated 7.3.2019

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. This document also shows how to use APEX Triggers as an alternate method for advanced developers. Common objects to trigger off of are:

Lead/Contact – Common use cases are when leads are created or 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 and Surveys 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 three 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.

Merthod #3 – Apex Triggers: This is for advance developers that prefer Apex Triggers over PB's or Flows

Method #1 – Simple

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

1. For the immediate action - Set Create a Record to **Scheduled SMS** then set these parameters either via a Process Builder or Flow.

    a. Only these parameters defined below are allowed. You cannot set the **OWNER** nor the **Message_Text** fields even though the Scheduled SMS object will expose then. Method #2 does allow a manufactured string to be provided but Method #1 does not, you must use a SMS_Template_Id or a Question_Id

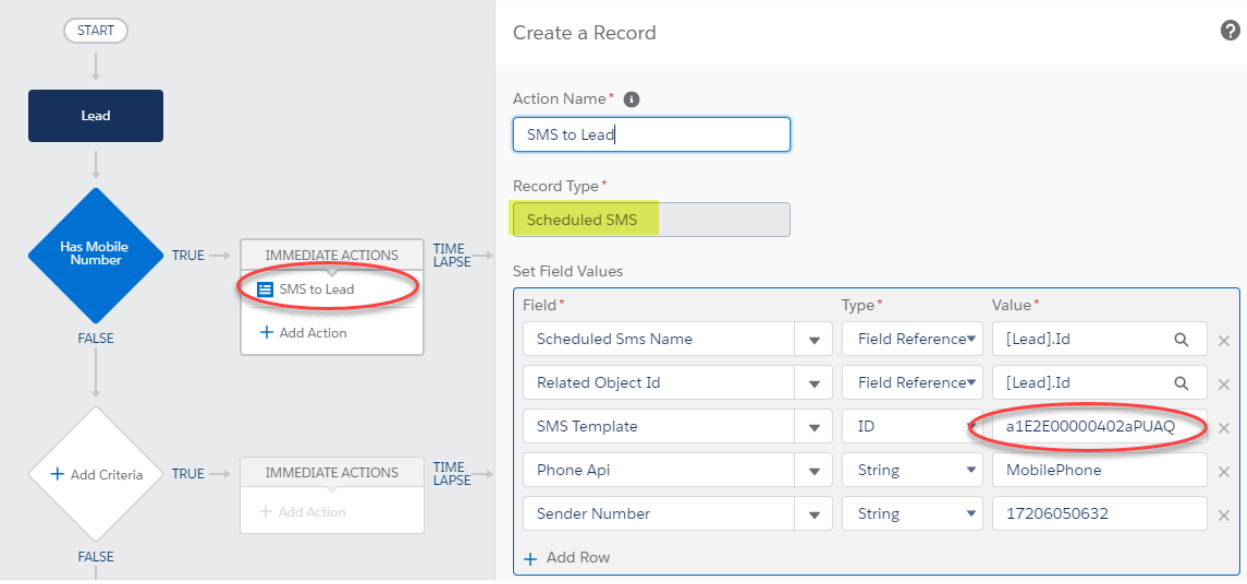


Figure 1 - Method #1 showing the 5 fields required for sending an SMS – note you can also set the QUESTION field which is the first question of the 360 SMS Survey object which will then trigger an automatic Question/Answer survey.

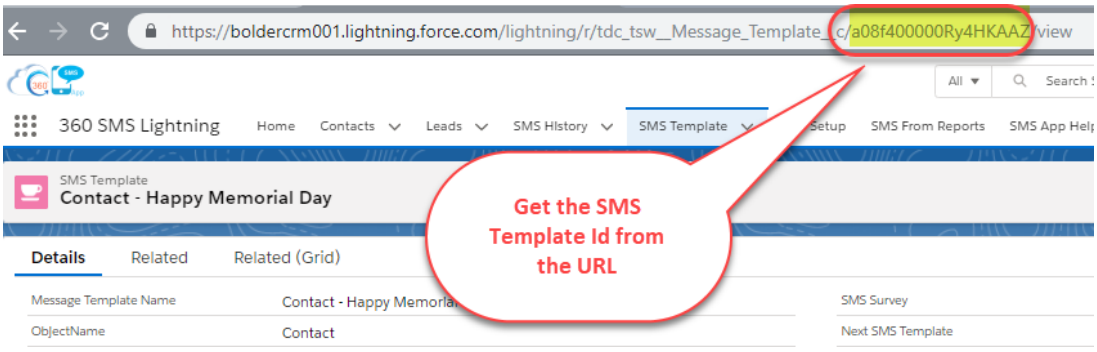
2. **Scheduled SMS Name**: Must be the ID field of the triggered object/record. This is used in conjunction with the SMS Template and must match by object in order for the merging to occur, e.g. Supply a Contact.Id and a matching SMS Template based on the Contact object.

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 an 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. **Phone Api**: Supply the phone field **API Name** to send the message TO. This can be a string such as "MobilePhone" or "Mobile_Phone__c" (custom field) or the actual phone value referenced from the record, i.e. Lead.MobileNumber but it can also be pulled from the Incoming SMS SENDER_NUMBER, set with a formula, or for SMS to employees you might use the User.MobileNumber.

a. Especially when using the **Scheduled Time** field below to schedule the SMS, we strongly recommend using the Field Name for the PhoneAPI value rather than the Field Reference because the App will dynamically get the phone value at the time it fires, whereas if you explicitly set the value via Field Reference but the MobilePhone field is blank at the time, your SMS won't fire even if you set the value. Or if the MobilePhone field changes in between the time that the message is scheduled versus when it Sends, then the old value would be used.

5. **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*



6. **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. We highly recommend utilizing the Survey object because most outbound SMS will likely generate a response which can now be automatically handled.

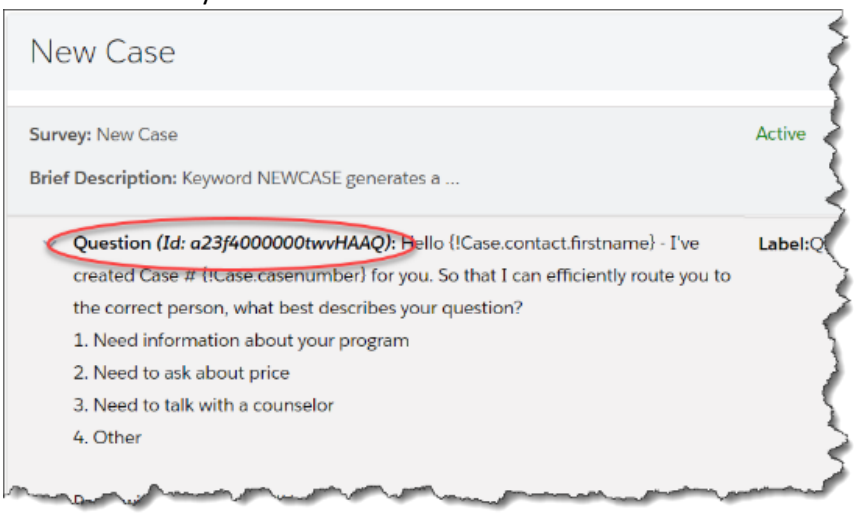


Figure 2 - Obtain the question Id from the first question of a Survey

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.

    a. The sender number must always be in format CountryCode+Number with absolutely no formatting or special characters, e.g. 17206050632. When dynamically setting the SenderNumber such as from Contact.Owner.Phone or from the Incoming SMS_History.To_Number use this valuable formula to strip out all the characters:

```
/************************************************************
Here's an example of getting the Sender Number from the
Contact.Owner.Phone field (USER object).
************************************************************/
'1' &
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE( Owner.Phone ,
"(",""),
")",""),
" ",""),
"-",""),
"+",""),
".","")
```

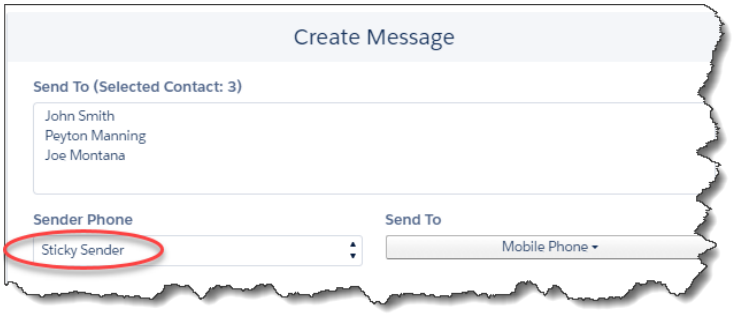
Figure 3 - Useful formula for reducing a number to 17206050632 format

    b. We recommend using the **Sticky_Sender__c** field which is a custom formula field that the **Send SMS** buttons and **Conversation View** recognize and overrides the users Default SMS Number with the formula value. The idea is that the number "Sticks" to the record so the customer always receives SMS from the same number.

        i. The formula field allows one to define their own business logic for record-based sender numbers so the customer always receives SMS from the same number.

        ii. This is useful for geography-based solutions (USA vs. UK numbers) or when each Record.Owner has their own sender number.

iii. Sticky_Sender is especially useful for Batch Texting so a marketing user can batch text many customers across many numbers.



8. **Scheduled Time:** The scheduled time field can optionally be set. If a value is provided (usually with a formula), then instead of the SMS going out immediately it creates the Scheduled SMS row as a related list item and fires the message at the allotted time.

   a. If using this feature, you MUST also set the Lookup field of the object whose Related List it should appear under, such as Contact or Lead. Setting the **Related Object** field is not the same as setting ContactId or LeadId, that field has a different purpose.

   b. Usually you'll use a formula such as these common ones below. Remember, Salesforce date math is in DAYS so you might Google your desired syntax for minutes and hours.

      i. NOW() + 1 /* Same time as today + 1 day, i.e. tomorrow */

      ii. NOW() + (1/24) /* 1 hour from now */

      iii. NOW() + (10/60/24) /* 10 minutes from now */

      iv. Specifying a specific time of day can be tricky because Salesforce stores time in Greenwich Mean Time (GMT) which you must account for. Below is an example of setting a Happy Birthday SMS to go out at 9:00am Mountain Time which is -7 hrs from GMT, so we set the time as 16:00:00 GMT.

```
/* Schedule Birthday wish for this yr 16:00 GMT = 9:00 MDT */
/* DateTime has to be in format:  YYYY-MM-DD HH:MM:SS      */
DATETIMEVALUE( TEXT(YEAR(TODAY())) & "-" &
               TEXT(MONTH([Contact].Birthdate )) & "-" &
               TEXT(DAY([Contact].Birthdate)) & " 16:00:00"
             )
```

   c. More on Scheduled SMS with this document: Triggered Scheduled SMS w/ Process Builder

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. This can be very useful as the REPLY comes back into whatever the RELATED OBJECT is. Use cases:

1. Linking a message to the parent CONTACT object when the message is initiated from a Salesforce EVENT (event reminders) or some other child object.
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.

Read the related section named "Roll-up SMS Conversations to Parent Objects" to learn a technique where you allow the SMS to be sent from the child object but then with a process builder you also set the Ids of the parent Contact and/or Account objects.

Below is an example which triggers an internal user notification SMS 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.

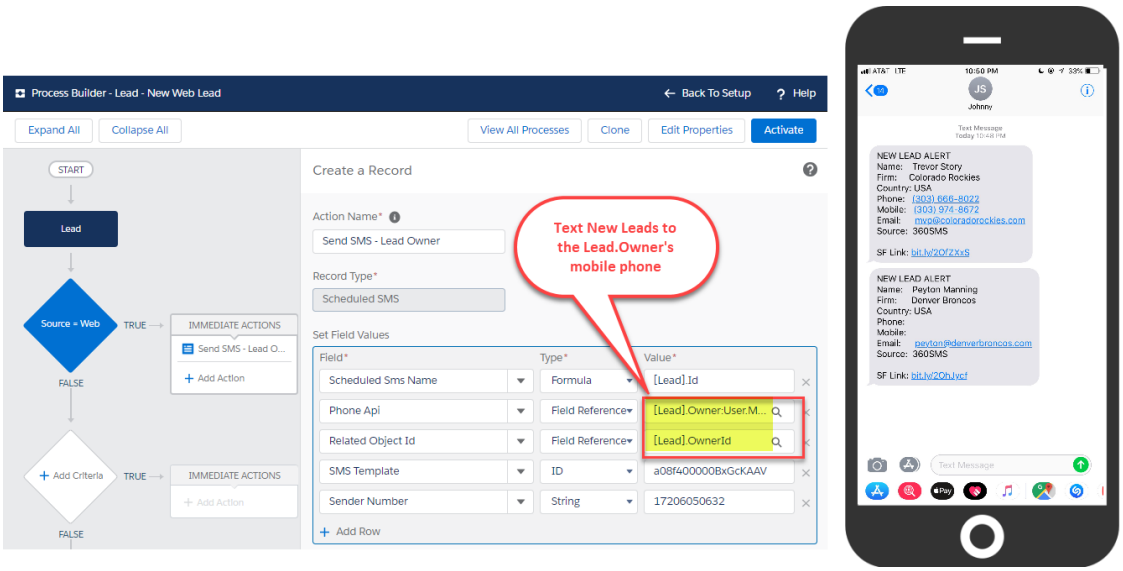


Figure 4 - Typical internal alert via SMS sending to an employee but triggered from the Lead creation

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.

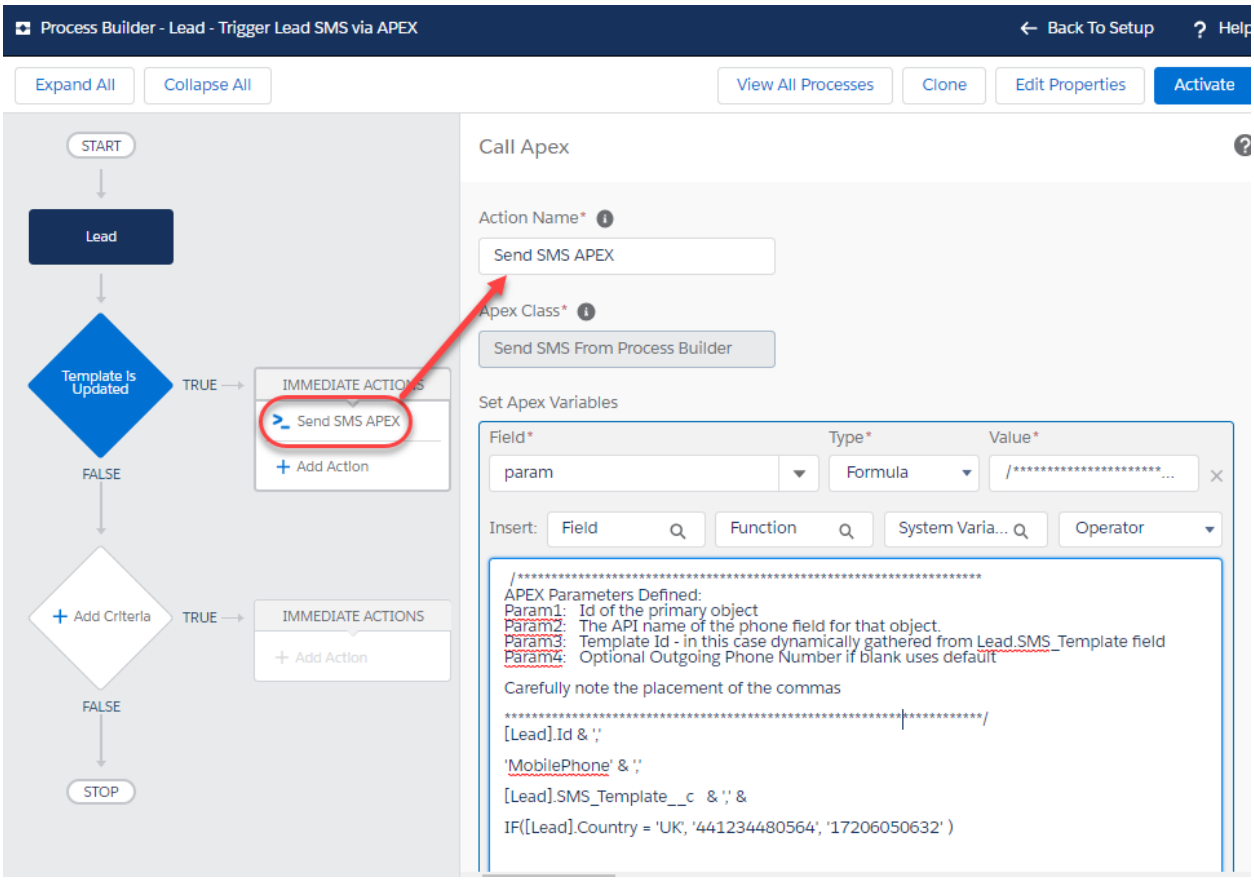


Figure 5 - Code example of sending regular SMS via the "Send SMS From Process Builder" Apex class

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'
```

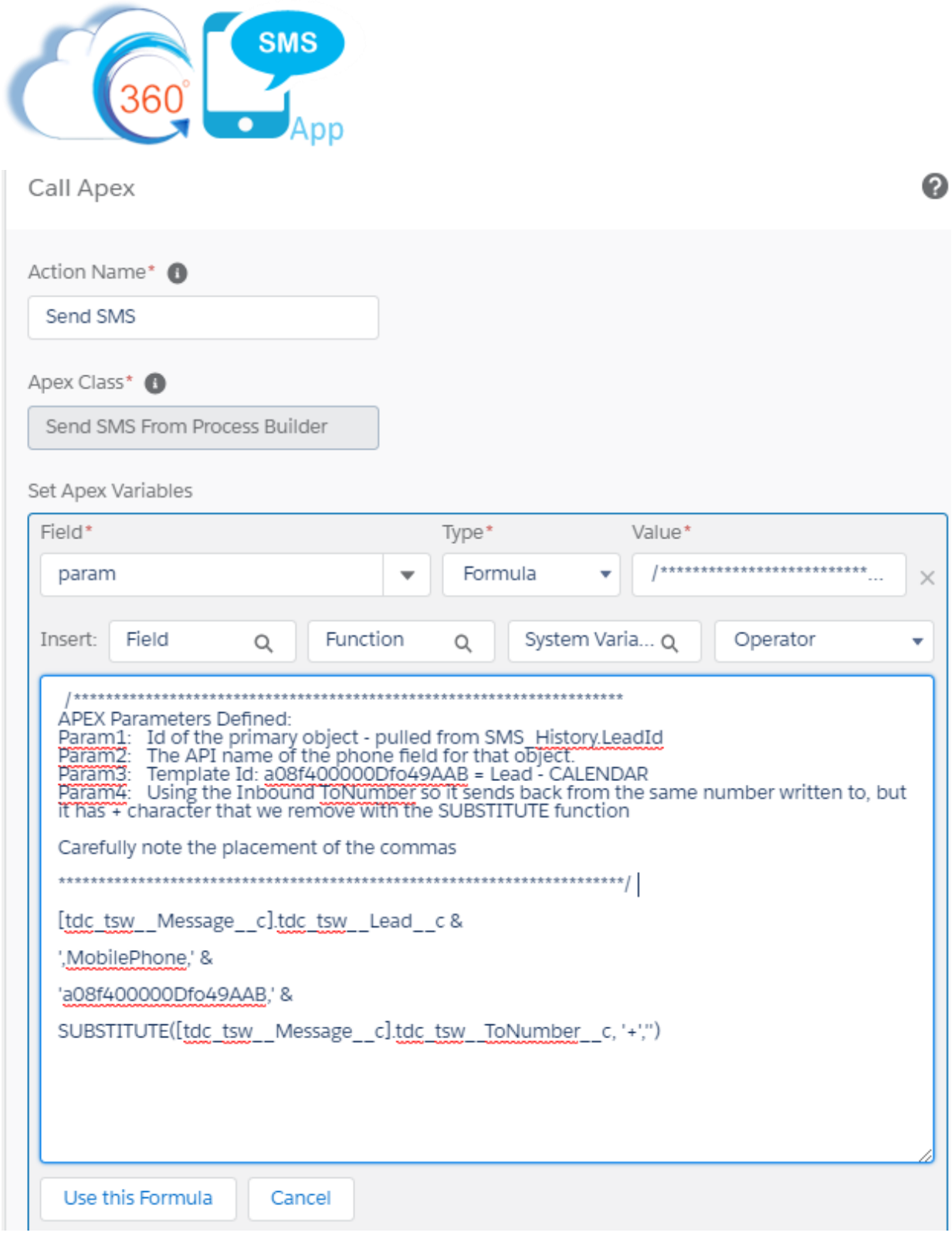


Figure 6 - Example APEX when triggering a reply from an Incoming SMS History

Method #3 – Apex Triggers

For more advanced developers that prefer using APEX Triggers over process builder, one can simply call the APEX class named **tdc_tsw.GlobalSMSSender.sendSmsWithPhoneAPiAndTemplateId** from your code.

The class accepts three parameters:

1. **Mobile Phone Field:** The field where the phone number of the contact would be stored, e.g. **objContact.MobilePhone**
2. **Template Id:** The TemplateId similar to the methods described above, e.g. **'a022800000Pj0Ia'**
3. **Object Id:** The id of the record which is being sent the SMS, e.g. **objContact.id**

Below is a sample trigger from the CONTACT object.

Trigger (for Contact object)

```
trigger ContactTrigger on Contact (after insert) {
      if(trigger.isAfter && trigger.isInsert) {
          ContactTriggerHandler.onAfterInsertContact(trigger.new);
       }
}
```

Apex Class

```
public class ContactTriggerHandler {
    public static String onAfterInsertContact(List<Contact>
lstContact) {
        String result;
        if(lstContact.size() > 0) {
            for(Contact objContact : lstContact) {
               if(objContact.MobilePhone != null) {
                        try {
      tdc_tsw.GlobalSMSSender.sendSmsWithPhoneAPiAndTemplateId(
      objContact.MobilePhone,
      'a022800000Pj0Ia',
      objContact.id);
                                result = 'Success';
                    } catch(Exception e) {
                                result = 'failure';
                    }
                }
            }
        }
        return result;
    }
}
```

Note that there are other APEX methods available as well which are intuitively named:

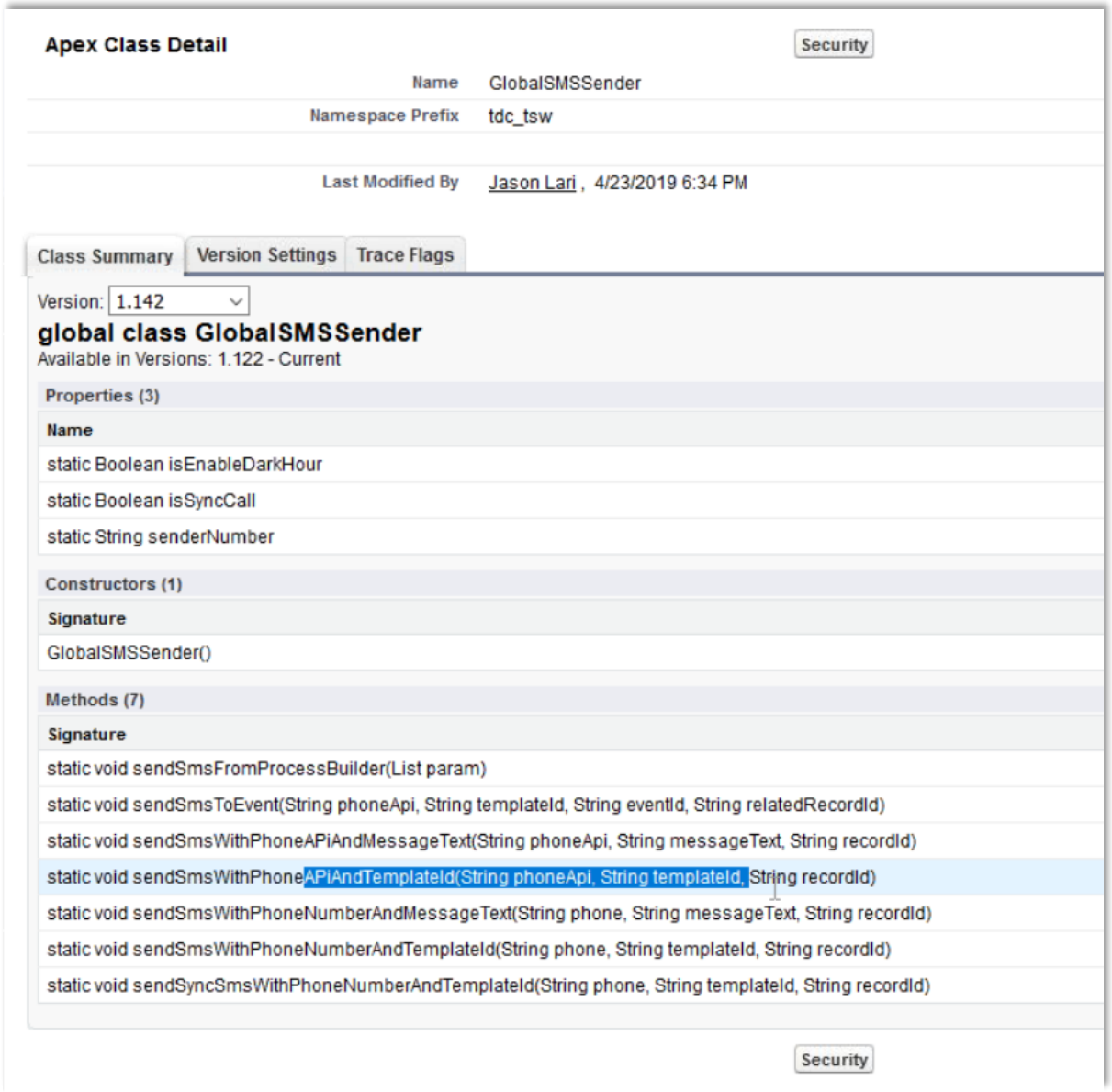


Figure 7 - Additional APEX methods that be called via APEX code

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 9*.

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.

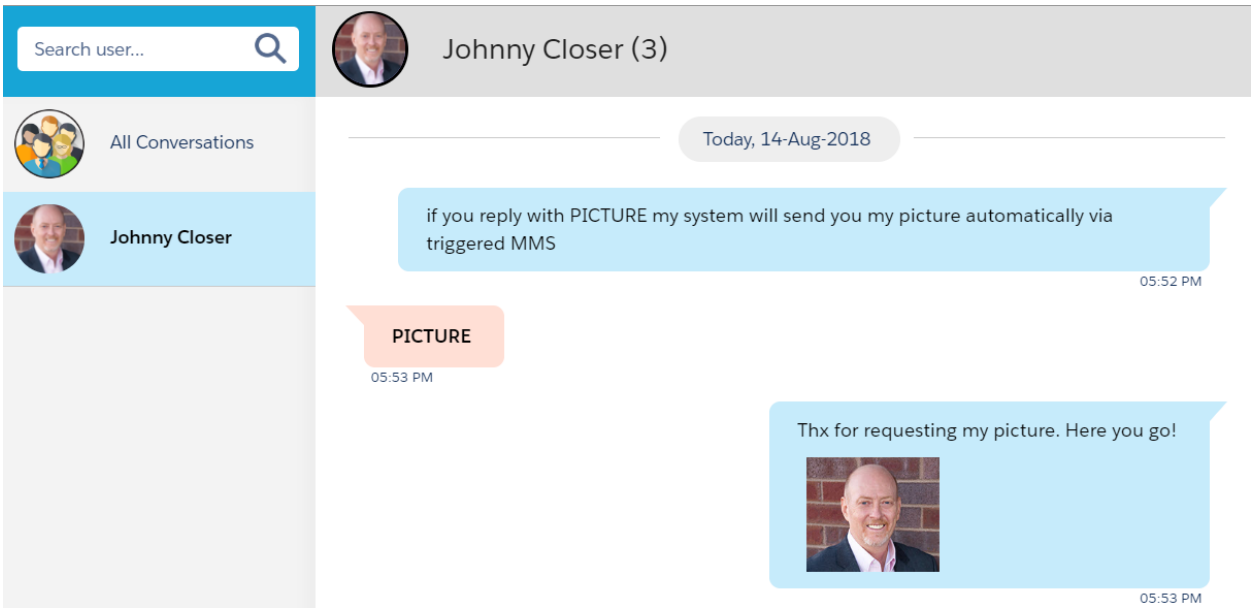


Figure 8 - MMS automation with keyword "Picture" sending pic of the Lead.Owner

Figure 9 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.

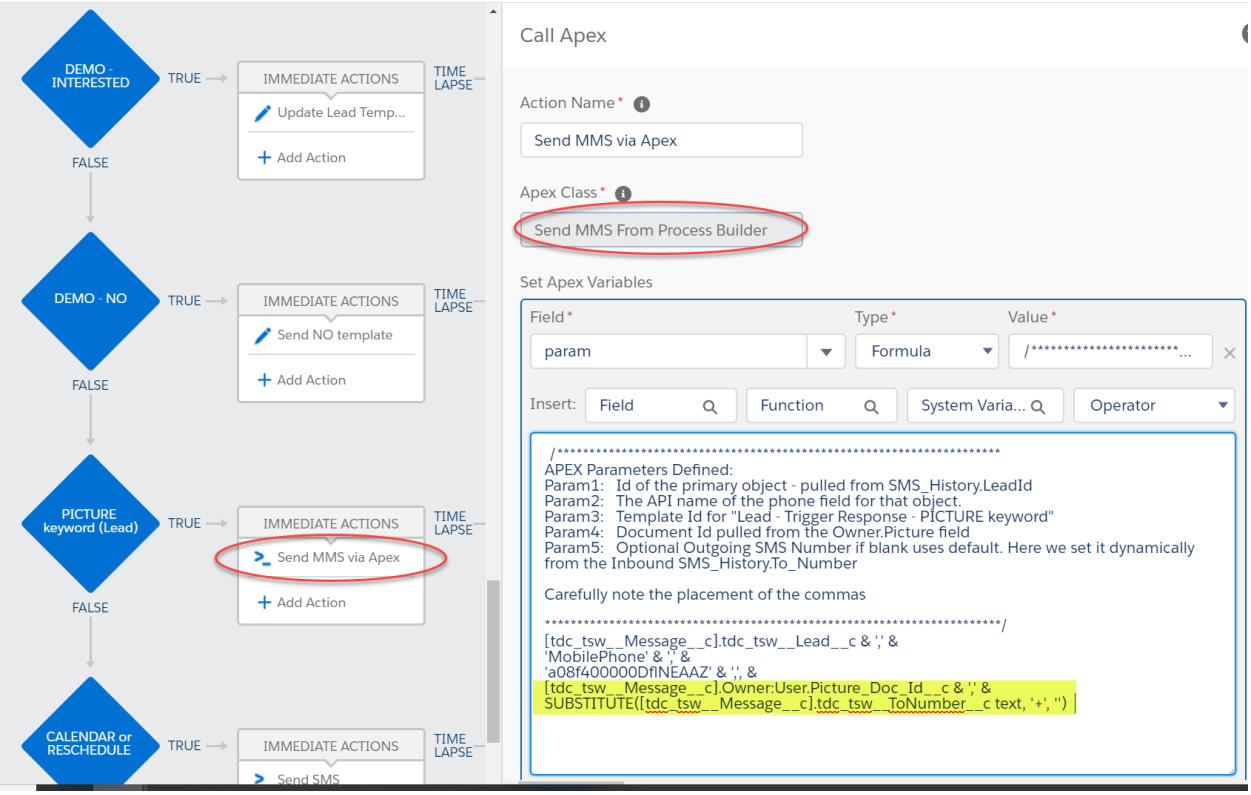


Figure 9 - PB code for executing triggered MMS via the "Send MMS from Process Builder" Apex class – this example is triggering off an inbound message with keyword "PICTURE". We obtain the picture by traversing up to the USER object and accessing a custom Picture_Doc_Id field. We use the SMS_History.To_Number to send back the reply.

Dynamic Sender Number

No matter which method one uses, it is common that the developer wants to dynamically set the outbound Sender Number so that the customer experience is to interact from a single phone number. This only applies to organizations with multiple sender numbers.

The sender number must always be in format CountryCode+Number with absolutely no formatting or special characters, e.g. 17206050632. When dynamically setting the SenderNumber such as from Contact.Owner.Phone or from the Incoming SMS_History.To_Number use this valuable formula to strip out all the characters:

```
/************************************************************
Here's an example of getting the Sender Number from the
Contact.Owner.Phone field (USER object).
************************************************************
/
'1' &
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE( Owner.Phone ,
"(",""),
")",""),
" ",""),
"-",""),
"+",""),
".","")
```

Figure 10 - Useful formula for reducing a number to 17206050632 format

There are four common use cases:

1. **Record Based Sender Number** – assign a specific sender number based on business logic such that all SMS to that Contact or Lead always comes from a single sender number whether that be Record.Owner based, Geography based or defined by other business logic.
2. **Dynamically set the Sender Number based on record owner** – pulling from the <record>.Owner's designated SMS number.
3. **Dynamically set the Sender Number based on geography** or some other business logic.
4. **Automatically replying to an incoming SMS using the number which the customer wrote to**, the To_Number for an SMS_History of type = Incoming (API name: `tdc_tsw__ToNumber__c`)
    a. Note that the 360SMS Survey tool does this automatically – always responding to the original number that the customer wrote to.

Record based Sender Number (Sticky Sender)

360 SMS allows one to create a custom field named specifically **Sticky_Sender__c** (you can label it any way you like). This is a custom formula field that can hold any business logic that one wants, such as Geography-Based numbers, Marketing Campaign Based Numbers or Record.Owner based numbers or it could be defined as the Last SMS number used for the contact. The point is that any business logic can be defined.

If this field exists, all interface elements of the platform (Buttons and Conversation View) will recognize that the Sticky_Sender and override the current users Default SMS Number and instead offer this field as the default. For example, a marketing user may want to select a large list of contacts for batch SMS but want the Sender Number to be the number that the customer has already been getting messages from, such as the [Record].Owners sender number.

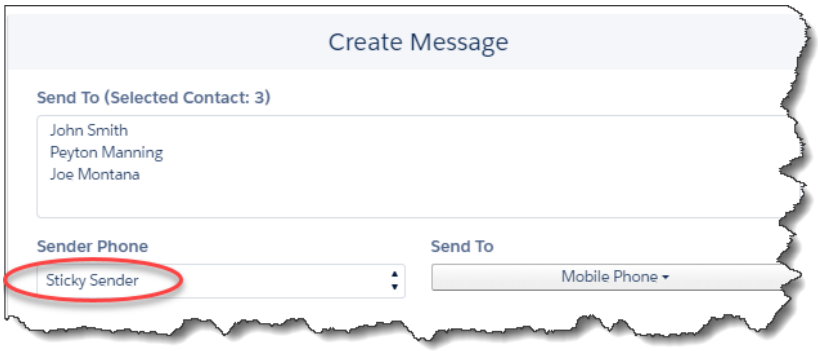


Figure 11 - When a sticky sender formula field is defined - it will default as the Sender Phone for batch, single and convo view

Programmatically, one references the Sticky Sender number field to make sure messages are coming from the correct number and to concentrate the business logic in the formula field.

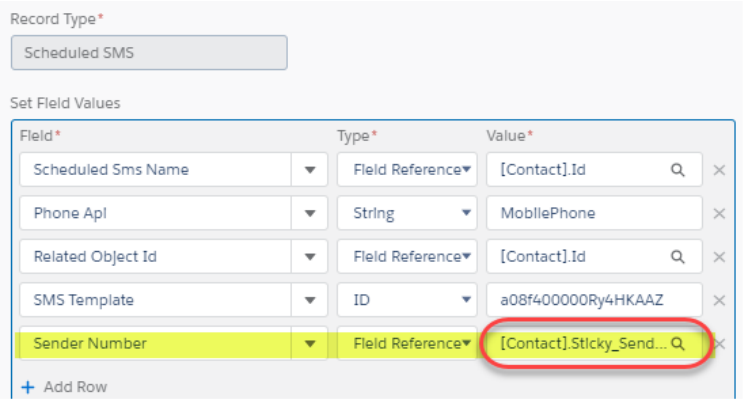


Figure 12 - Use the sticky sender field for your Sender Number to be consistent and concentrate your business logic in the formula field

Dynamic Sender Number – Record Owner

Many orgs use separate numbers for each user, primarily for voice call forwarding situations. The mapping of Users to Numbers is defined in the SMS Setup → User Configuration which is unfortunately 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 Sender 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. Be careful if using the standard User.MobilePhone or Phone field as Salesforce formats these numbers, such as (720)605-0632. The number needs to be completely unformatted and have the country code prefix, i.e. 17206050632.

You can now traverse to the User table such as Lead.Owner/Contact.Owner and get the number from your custom field. Or better yet define this logic in a Sticky_Sender__c formula field such as: Owner.SMS_Number__c or using the formula below which makes sure to strip out the various phone formatting characters.

```
/*************************************************************
Here's an example of getting the Sender Number from the
Contact.Owner.Phone field (USER object).
*************************************************************/
'1' &
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE( Owner.Phone ,
"(",""),
")",""),
" ",""),
"-",""),
"+",""),
".","")
```

Dynamic Sender Number – Geography or Business Rule Based

In larger organizations with many geography's or complex business rules it is desirable or even a requirement to use specific sender numbers. In the case of multiple countries, it is a requirement to use a sender number that matches the country of destination. There are three solutions here:

1. Use the Sticky Sender custom formula field explained above.
2. Utilize the 360 SMS Co-Pilot feature which will use its Area Code/Country Code picker to dynamically pick the sender number if you have purchased a matching Area Code or Country Code number in your pool. For more on Co-Pilot refer to this hyperlinked document: *Batch Texting - Auto-Routing*
3. Use the Process Builder formula editor to build in your business logic for the Sender Number but still you are better off now putting your logic in the Sticky Sender formula field rather than defining the logic in the process builder.

Dynamic Sender Number – Incoming SMS

The third 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 Sender 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
```

Dark Hours

360SMS has a feature titled ***Dark Hours***, which when enabled, delays any Triggered or Scheduled SMS messages from firing until the "Sending time/Next Day." In the screen capture below any messages that were triggered after 4:00pm and before 6:00am will be delayed until 6:00 AM of the following day.

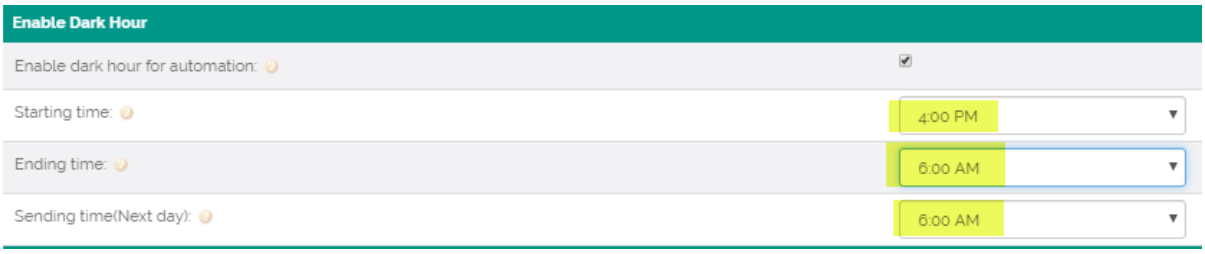


Figure 13 - Dark Hours enabled - any triggered messages will be sent the next day at the given time.

Technically what occurs is that if a process triggers the sending of an SMS during the dark hour time frame, it will create a Scheduled SMS related list record shown below with the Schedule Date/Time feature. Then the Scheduled SMS will fire the next day at the start time.

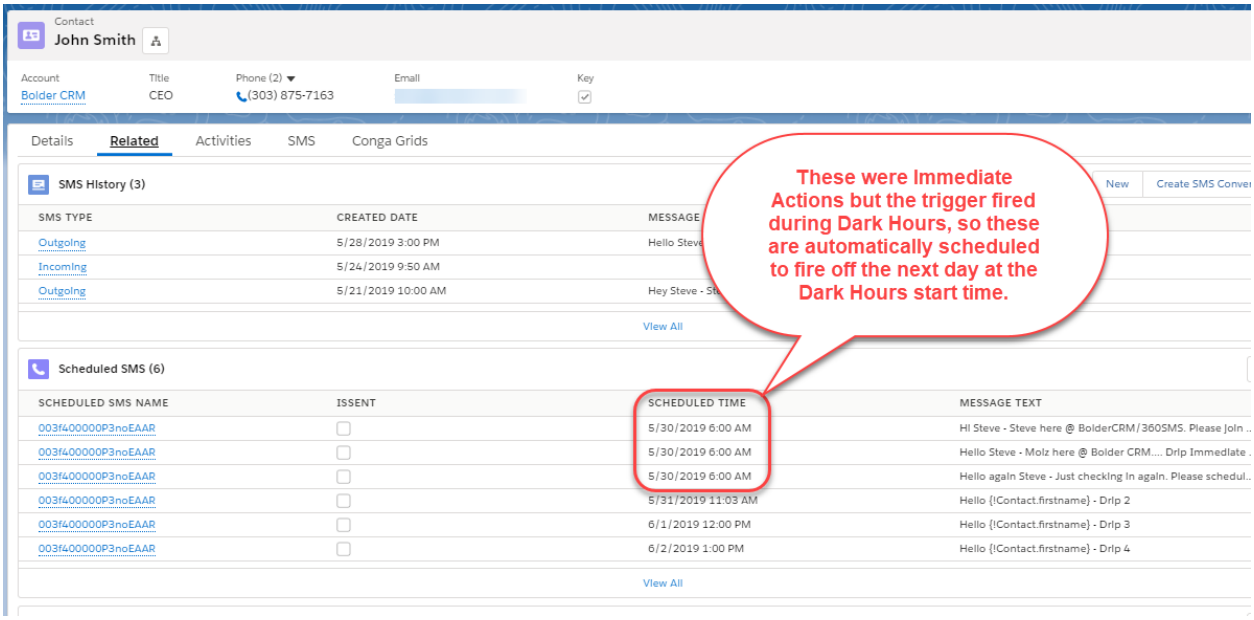


Figure 14 - Triggered SMS during dark hours creates a Scheduled SMS record with the Scheduled Time = Sending time//Next Day

Roll-up SMS Conversations to Parent Objects

It's good CRM best practice that when texting either manually or via automation from a child object, that the SMS "rolls-up" to also display on the parent CONTACT object. Or in the case of texting from the Contact it should roll-up to the Account so from the Account page one can see all the conversations with all the contacts.

The CASE object automatically rolls-up any SMS linked to the Case to the parent Contact so that when looking at the Contact record you can see contextually each conversation as it is related to both the Contact and the Case. For other objects we need to intercept the SMS History via Process Builder and do the roll-up with an Immediate Action updating the SMS History.

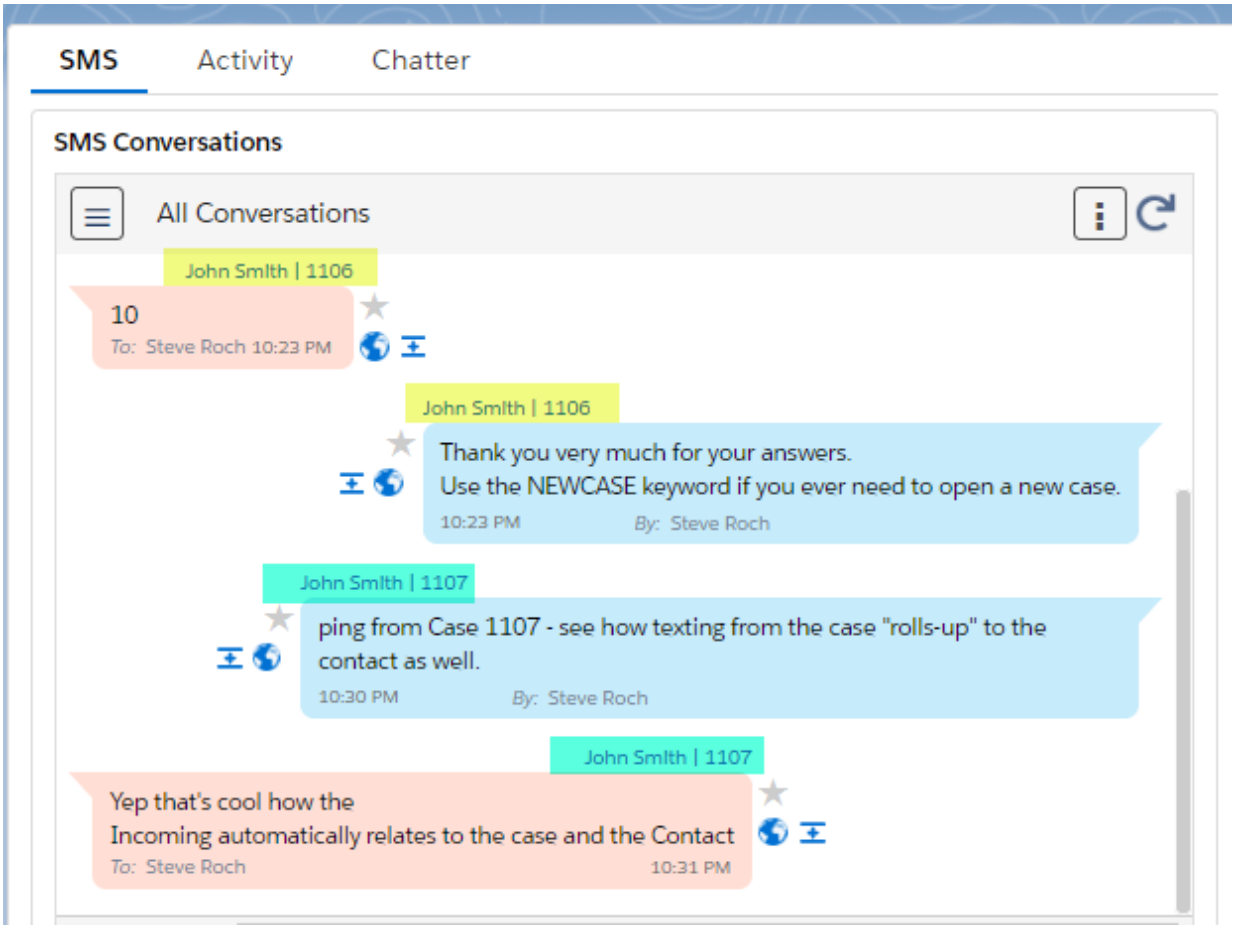


*Figure 15 - Texting from different cases automatically "rolls-up" the conversation to the Contact as shown here for cases **1106** and **1107** – we want to emulate this same functionality for other child objects.*

In the example below (*Figure 16*), we have a classic child object named Job_Listing which is the child of a Job and the child of a Contact (a contacts potential interest in a Job that is being offered for a recruiting company). Usually recruiting firms batch text or trigger text from the Job_Listing and they want to see

all the SMS from all the Contacts in the Job's SMS Conversation View but also from the contact record they want to see all the Job related texts they've sent the Contact (similar to *Figure 15* with Cases). This is the perfect scenario for a ROLL UP. We detect that the SMS History's primary related Object = Job Listing and then because the Job Listing is the child of a Job and a Contact we simply update the SMS History with those values, setting SMS_History.ContactId = Job_Listing.ContactId and SMS_History.Job = Job_Listing.JobId

Note that the platform uses the primary Related Objects NAME field for the **Sender Name** (the link that appears above the message and which shows in the Incoming Alert). It's a good practice when texting from Child objects to modify the Sender Name with a process builder so that in this case it reads "John Smith | SF Administrator", i.e. `Contact.Name & " | " & Job.Name`

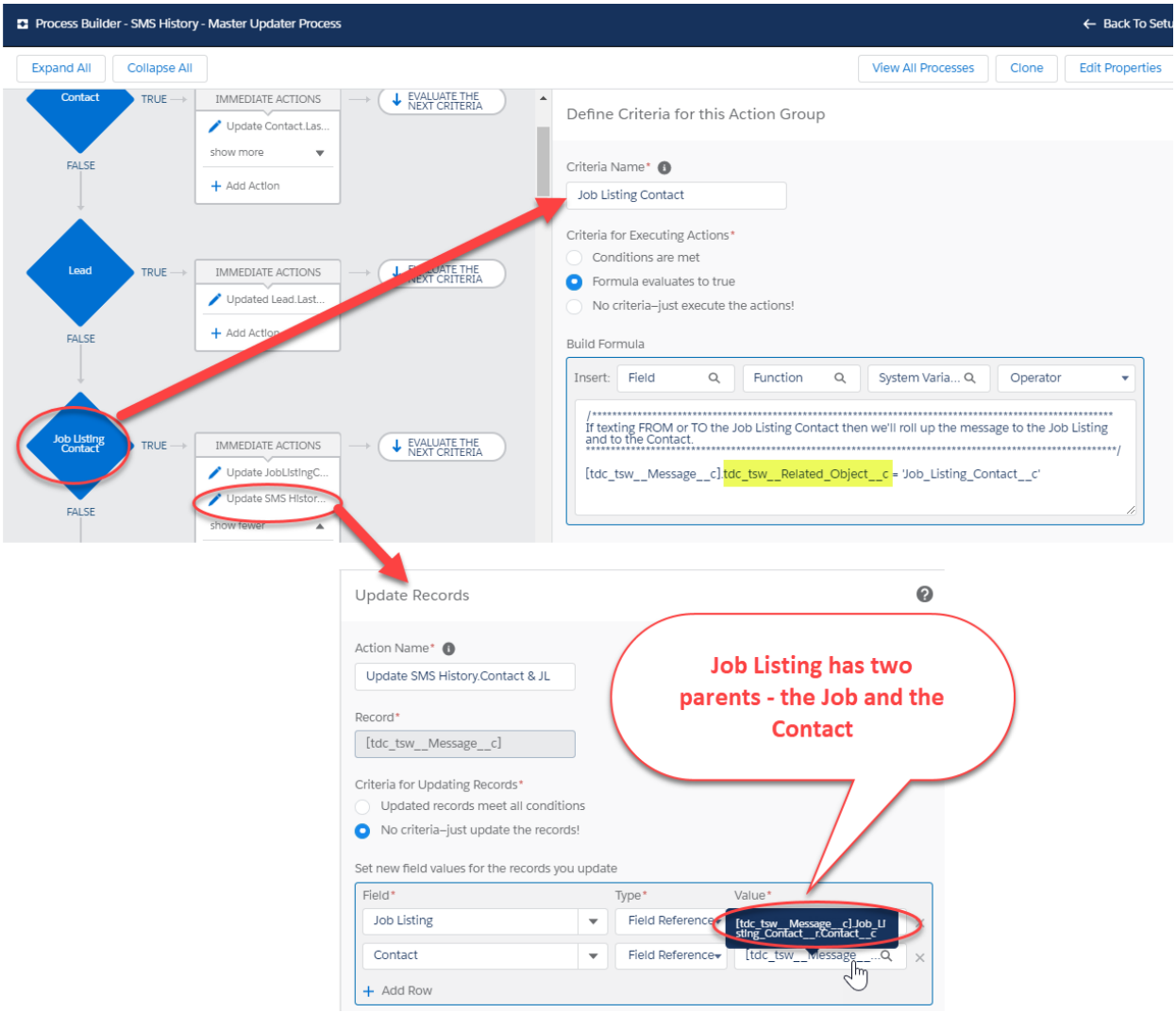


Figure 16 - Classic Roll-Up from a child object to the Contact – set the SMS_History.Contact Id = Job_Listing.Contact_Id

Create a Contact/Lead Last_SMS field

Often customers want to use Salesforce List Views based on the Contact, Lead or some other custom object to determine which customers have or have not received an SMS lately. However, because Salesforce List Views do not allow Cross-Object queries and even Salesforce Reports do not allow for a Does Not Exist clause this can be a challenge.

To resolve this Salesforce limitation, we recommend placing a simple Salesforce Process Builder in place which updates a custom field such as Contact.Last_SMS or Lead.Last_SMS whenever any SMS History record is created. At your discretion, you could also differentiator Incoming vs. outgoing by having fields such as Last_SMS_In and Last_SMS_Out but we usually find a single field to be sufficient.

Create a Process Builder on the SMS History such as the "SMS History – Master Updater Process" mentioned in previous sections. Then simply detect that the SMS History.Contact or SMS_History.Lead value is not null. Then update the respective records Last_SMS field with the SMS_History.Create_Date.

Now you can use the Last_SMS field in your Contact/Lead List View queries.

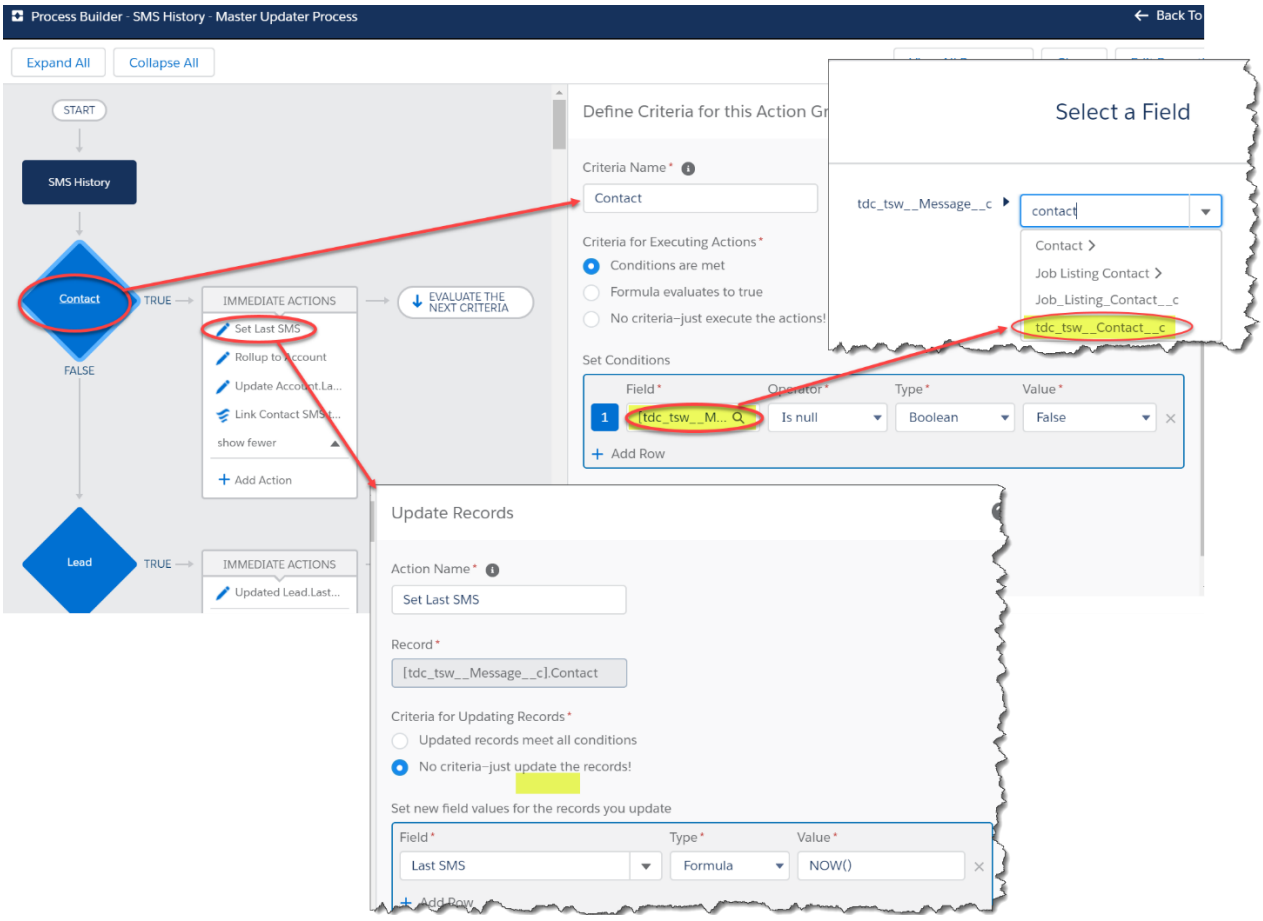


Figure 17 - Set a Contact.Last_SMS field

Triggered Texting to Internal Users

Some customers want to use the power of texting to switch 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.

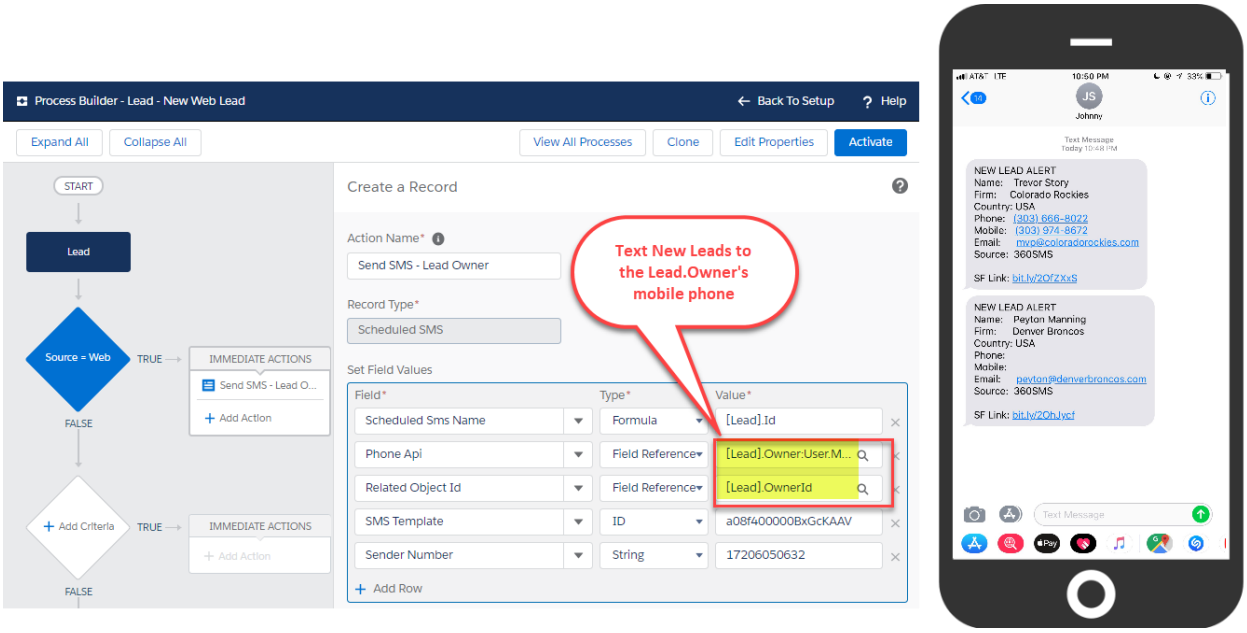


Figure 18 - Sending text alerts to internal users dynamically using the Lead.Owner.User.MobilePhone

Create a Master Send SMS Handler

If you anticipate creating using Method #1 for many different triggers you might consider adding an **SMS_Template_Id** lookup field to a Lead, Contact or any custom object. Then as shown below 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.

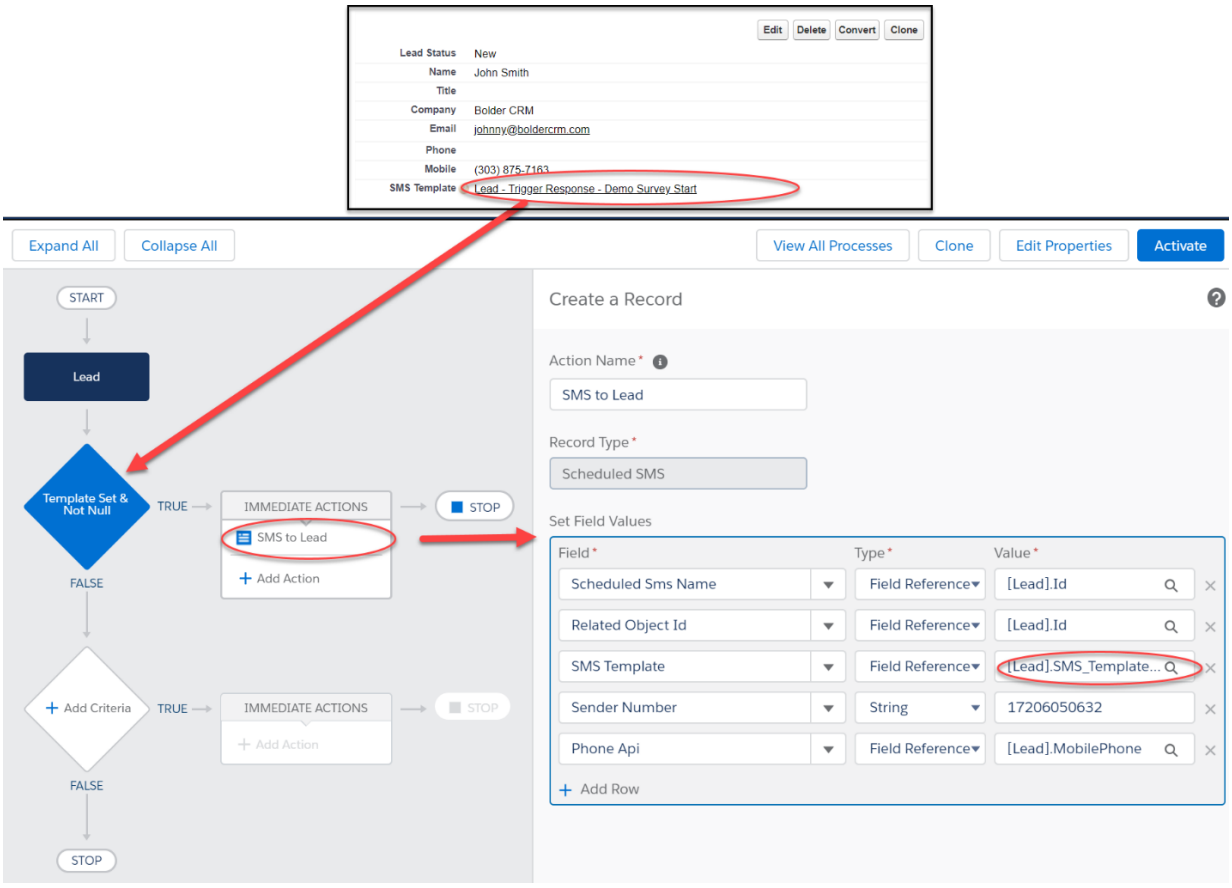


Figure 19 - In this scenario we are pulling the Template from a custom field we placed on the Lead for even easier automation. Now you can have other process builders that only need to set the Lead.SMS_Template and that alone will trigger an outbound SMS.

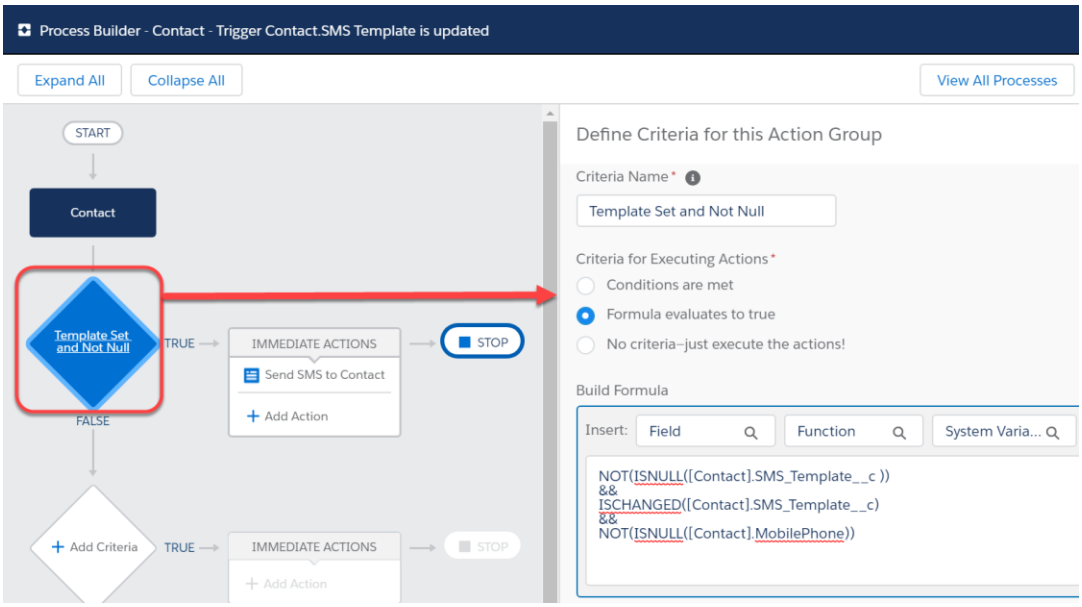


Figure 20 - When the SMS Template is changed - trigger the outbound SMS, Figure 2 shows the Immediate Action

Figure 21 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!

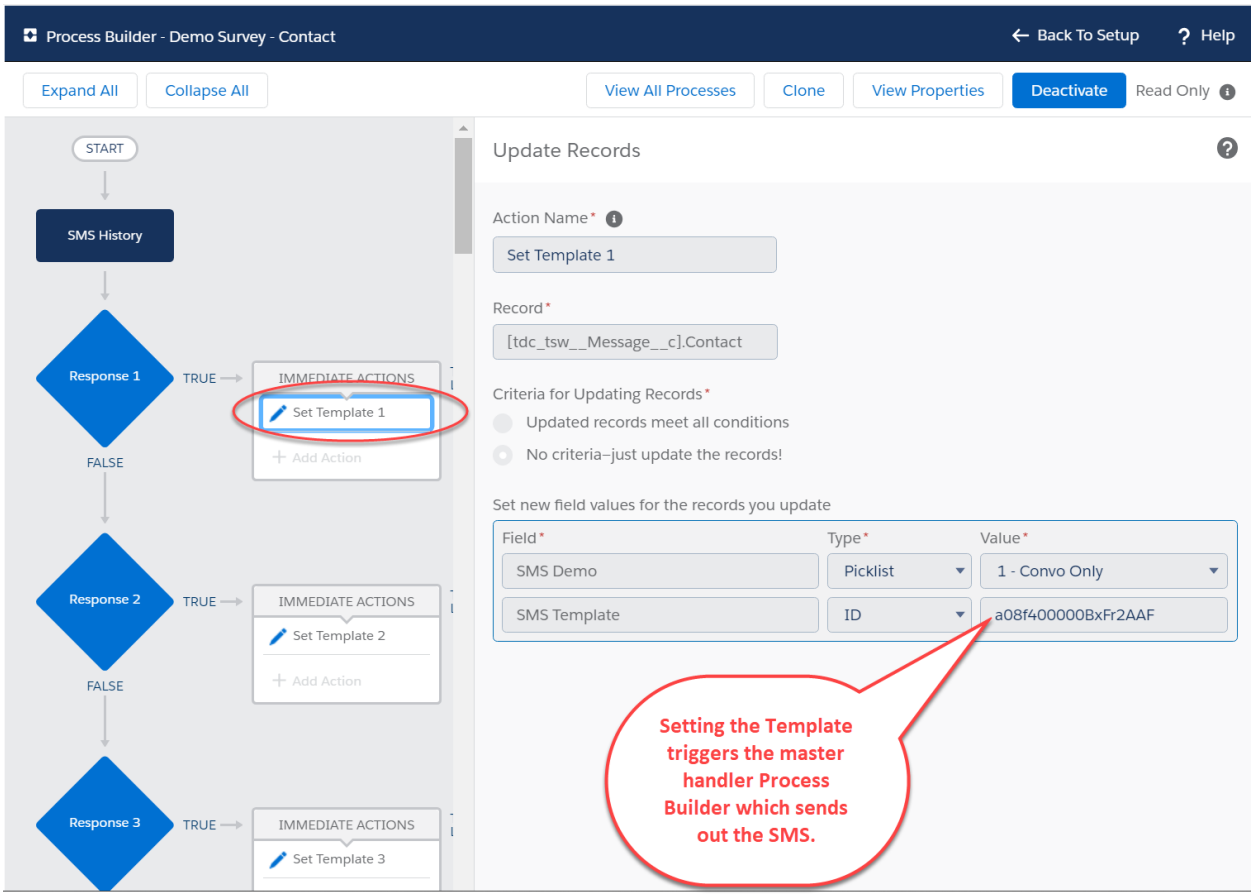


Figure 21 - Demonstration of the easier way of triggering an SMS via a change to the Contact.SMS_Template_Id (custom formula and matching PB)

Drip Campaigns

As a value-add from the Bolder CRM + 360 SMS partnership, Bolder CRM has utilized the unique ability of the 360 SMS platform to programmatically Schedule SMS with Process Builders or Flows, to create the Drip Campaign feature. This solution is sold as a separate module. Read the full **SMS Drip Campaigns** documentation or **Watch the short video** to learn more.

Drip Campaigns allow end-users (rather than Process Builder developers) to define a series of SMS Templates or SMS Surveys (aka iText or intelligent ChatBots) to be scheduled at various intervals defined by a **Days Offset** field. One can also optionally define the time of day that a message goes out.

Drip Campaigns can be assigned/stopped manually using the **Drip Campaign Applied** related list or programmatically with a single process builder command. Drip Campaigns can also be stopped with a single process builder command. More on those techniques below.

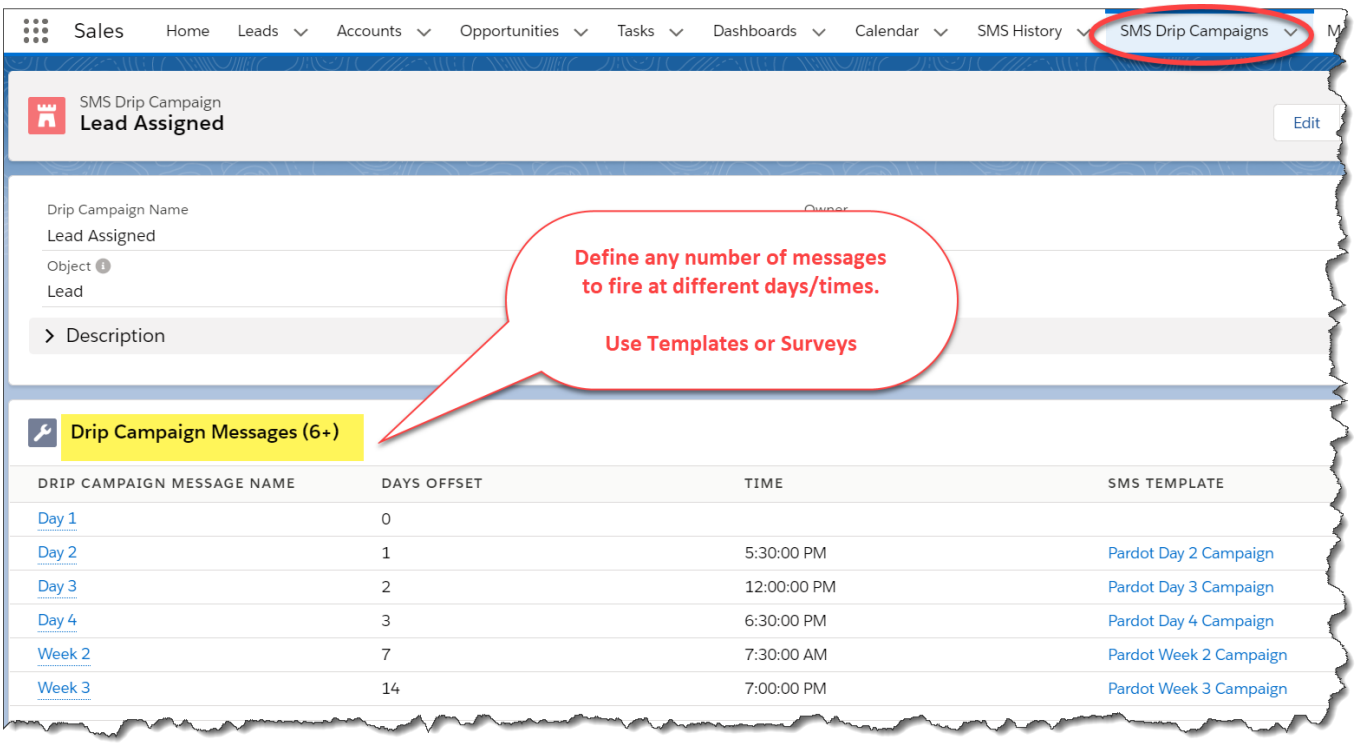


Figure 22 - Drip Campaign are defined by a series of SMS Templates or Surveys to fire at different Days and Times.

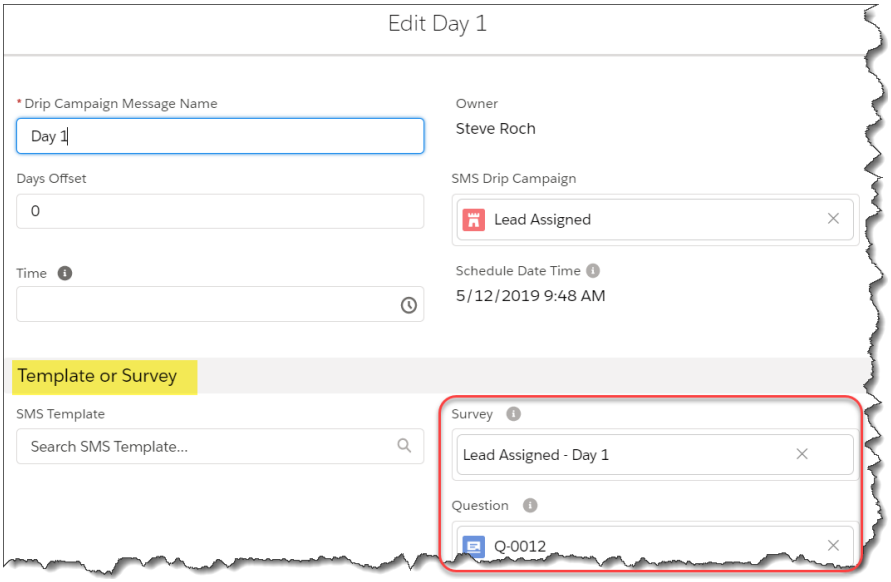


*Figure 23 – Define each days Template or Survey by defining the **days offset** from the day it is applied and an optional time of day*

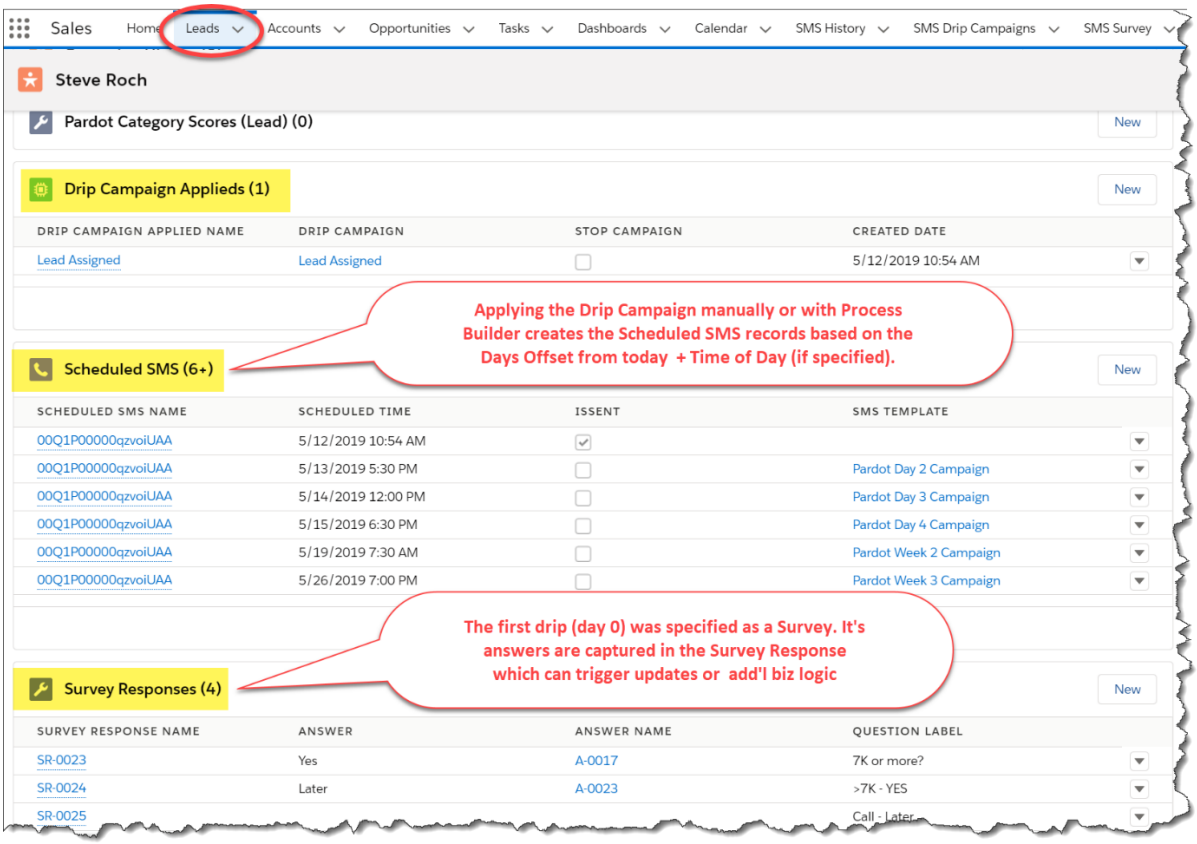


Figure 24 - Drip Campaign Applied creates Scheduled SMS records at the given Days Offset + Time of Day (if specified)

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 or take other actions using Process Builders on the SMS History or Survey Response object to inspect the incoming answer to a specific question. Read more in the document titled: Surveys and Incoming Keyword Processing

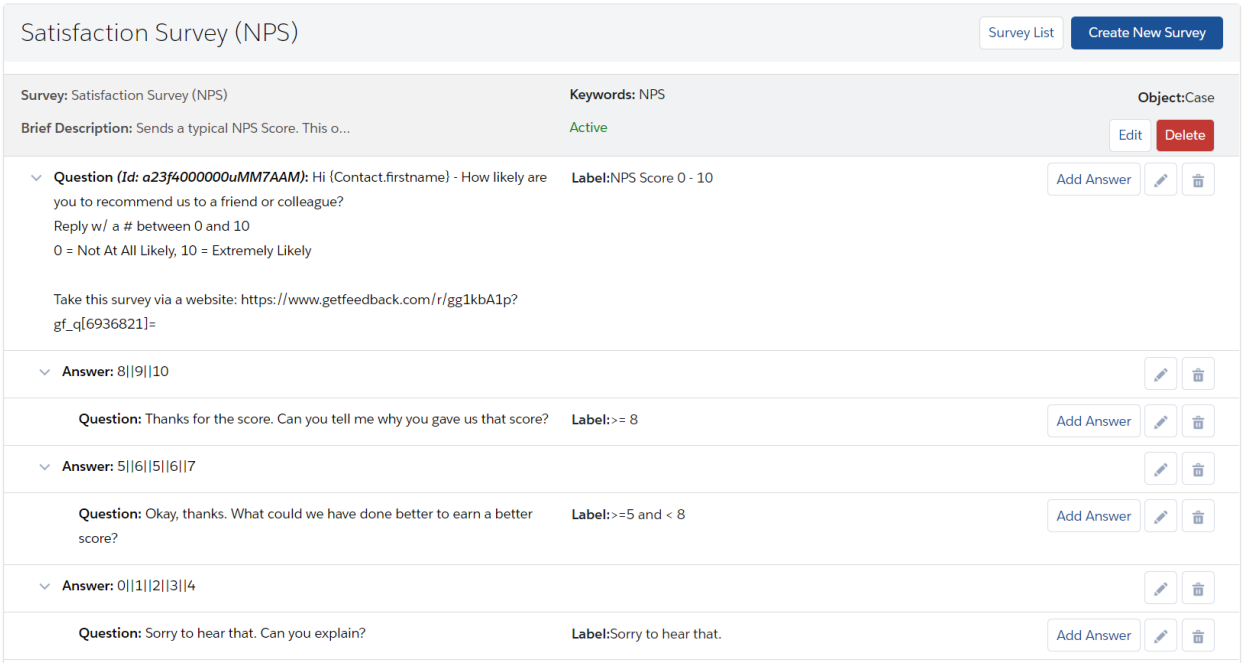


Figure 25 - Typical NPS score survey (Customer Satisfaction Survey)

Using Salesforce Flows

The Salesforce Flow technology is worth a short discussion as it provides considerably more power than Process Builder, such as the ability to add complex business logic, use variables, lookup records, loop through records, delete records and more.

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 #, can I get your email address so I can look you up by email?***"

In this 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 Email, Name 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 26 shows how a normal process builder can call a Flow passing in parameters that we gather from the SMS_History record. *Figure 27* then shows the details of the flow where it:

1. Looks up the email from the Contacts object using the Get Records object of flows
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 19* where we only update the Contact.SMS_Template field which in turns triggers our Master SEND SMS Handler.

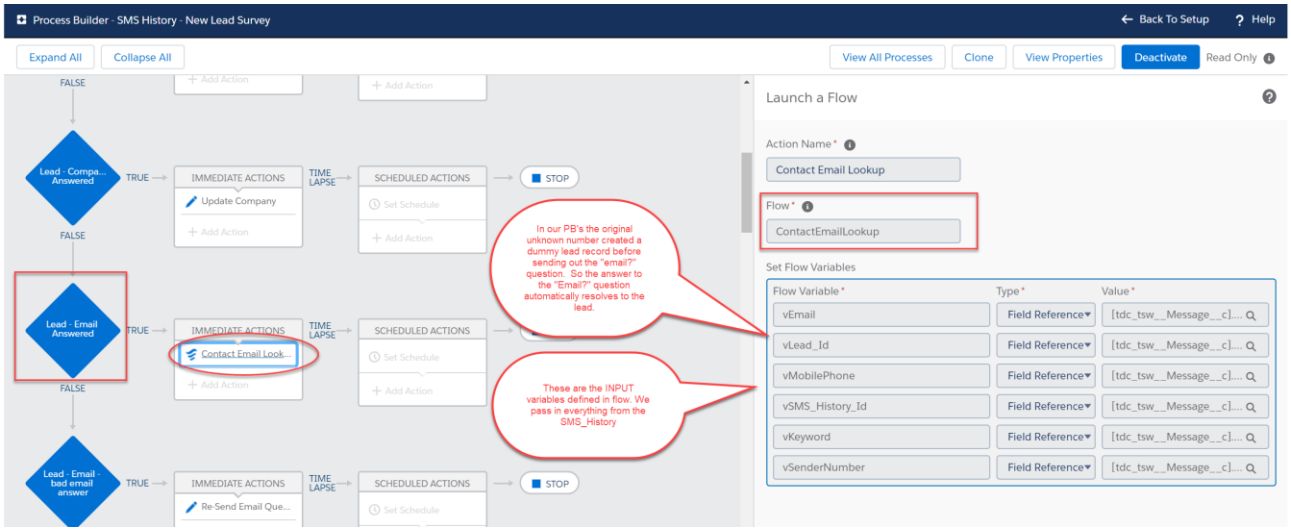


Figure 26 - Example of an SMS_History Incoming process builder triggering a call to a flow.

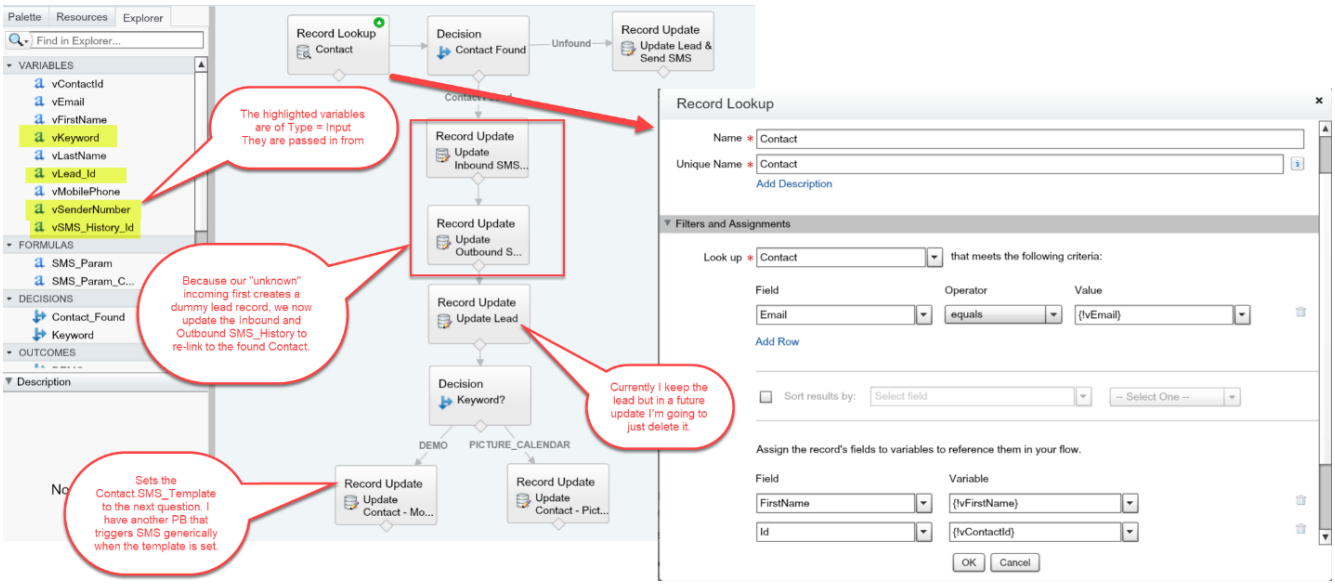


Figure 27 - A flow which does a Record Lookup based on a response from an inbound SMS with an email address

Troubleshooting SMS Process Builders

This section covers some basic troubleshooting techniques for triggered SMS. Debugging process builders and flows is intense and tedious work involving the customers business logic and thus even though it is common for the customer to think the platform is misbehaving there is always an answer in the coding or with the Salesforce users that are triggering the process. As a result, technical support cannot debug process builders. Consulting agreements are available though.

1. Always assume that your SMS isn't firing because of a problem with your business logic, its parameters or with the user triggering the process. You'll save yourself a lot of grief focusing your efforts on these items rather than thinking the platform isn't working.

2. **Criteria** – Most issues around an SMS not firing are due to bugs or misunderstandings with the criteria within the process builder. First, remove or minimize your criteria to see if the SMS will fire without the criteria. Of course, this is not always practical but somehow remove the criteria as the culprit first.

3. **Use the Scheduled Time feature** – A fantastic troubleshooting technique is to set the Scheduled_SMS.**Schedule_Time** field as documented in Method #1 above. Even if you want your SMS to fire immediately set the Scheduled Time field to formula NOW() or NOW() + (1/60/24) to add 1 minute from Now().

4. **Salesforce Scheduled Actions** – It is quite difficult to troubleshoot the Scheduled Actions of a Process Builder since you can't physically see whether or not your Scheduled Action has fired.

    a. Consider temporarily changing your logic to use the **Scheduled Time** field in method #1 to "schedule" the SMS as a Scheduled SMS related list item rather than using the Scheduled Action. This confirms whether your Criteria is the problem or something else.

5. **Object + Template** mismatches – Remember that when using Method #1, the Scheduled SMS Name value is the Id of the object which must match the object of the SMS Template parameter. Object Id + Template Id mismatches are common problems.

6. **Trigger User** – Many problems arise when the Salesforce user that is making the change or creating the record that triggers the Process Builder does not have a 360 SMS License or the user is not associated to the Sender Number in the SMS Setup → User Config.

    a. Check the Created By or Last Modified user of the record to make sure they are licensed and that the sender number being used is associated for the user.

7. **Integration Users** – commonly customers will have integration processes or website driven processes that create or update records under the context of the **Site Guest User**. Since this is not a traditional user it can create problems as the user cannot be assigned a license in the traditional way and the user cannot be associated with the Sender Number.

   a. Contact technical support as the workarounds for this situation are quite complicated.

8. **App User** – the system administrator that installed the app is the Created By user for all incoming SMS History records. Often the original system administrator will give up his license or will neglect to assign himself to all the possible sender numbers that might be used.

   a. You can reassign the SMS App User by having another Sys Admin user press the Incoming/Outgoing Setup button and changing the App User in SMS Setup → General Settings.

About the Author

Steve Roch, CEO of Bolder CRM is an SMS Industry expert having worked or consulted with the top three SMS Apps on the Salesforce AppExchange and also having built the popular Salesforce app **ActionGrid™**, acquired by Conga in April-2016. Bolder CRM is the exclusive distributor of 360 SMS in the United States, Canada and the United Kingdom.

Learn more about Steve and Bolder CRM at https://boldercrm.com/360SMS and https://www.linkedin.com/in/steveroch/

Call or Text: 720.605.0632
Email: steve@boldercrm.com