Sending SMS w/ Process Builder & Flows

Updated 11.15.2022

Contents

Sending SMS with Process Builder & Flows Basics

Salesforce Process Builder and Flows 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 from are:

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

Birthday SMS – place a trigger on the Contact.Birthdate field to schedule an SMS automatically every year – read about this specific use case in Scheduled SMS with Process Builders

Meeting Reminders – place a trigger on the Salesforce EVENT object or custom Appointment objects such as with Salesforce Field Service. Read about this specific use case in the dedicated document named: SMS Event Reminders

SMS_History – Ownership/Notification overrides. Commonly, customers have complex business rules that dictate changing the default platform behavior of an incoming message inheriting from its preceding outbound (including the Owner field). Thus, customers place triggers on either the Outgoing, Incoming or both to re-route based on their custom business logic.

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 but for this case you should really just use the 360 SMS iText functionality (aka Surveys/Chatbots) which is a no coding approach to this sort of use case. Read more about iText in Surveys & Incoming Keyword Processing.

Survey_Response – Although 360SMS has a dedicated **Survey Action Handler** product now for taking common actions on iText/Survey question responses, one can certainly do it themselves with triggers on the incoming Survey Response object.

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.

Methods

There are three primary methods of triggering an outbound or scheduled SMS:

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

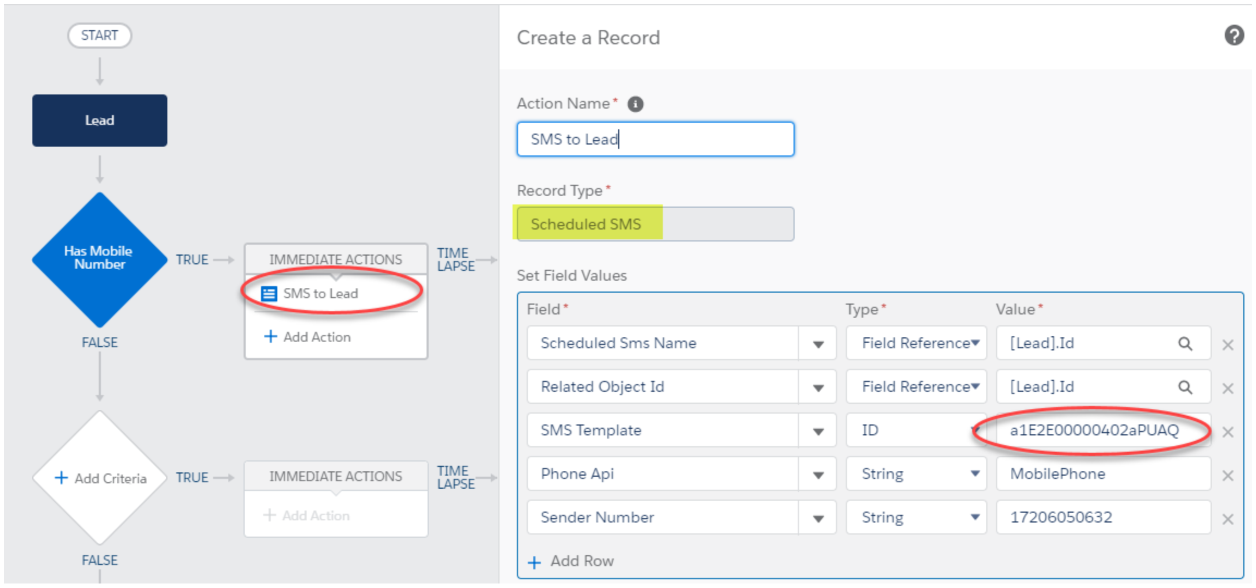
Method #2 – Apex Class: An alternate method that uses a single command with a concatenated formula string of parameters. Handy because it's copy/pastable but missing some of the richness of Method #1.

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:

First off, **Create a Record** to for the **Scheduled SMS** object then set these fields either via a Process Builder or Flow.   These are listed in order of importance, the first 5 being the minimum requirement.



Create a Record

Action Name*: SMS to Lead

Record Type*: Scheduled SMS

Set Field Values

Field*	Type*	Value*
Scheduled Sms Name	Field Reference	[Lead].Id
Related Object Id	Field Reference	[Lead].Id
SMS Template	ID	a1E2E00000402aPUAQ
Phone Api	String	MobilePhone
Sender Number	String	17206050632

Add Row

Figure 1 - The most important 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 (aka iText) which will then trigger an automatic Question/Answer survey. And you can set the Scheduled_Time, Attachments. Look to the flow example for more possibilities and we strongly recommend Flows over Process Builders.

1. **Scheduled SMS Name**: Must be the record ID field of the triggered object/record and this Record ID's object must match the object of the SMS Template or Survey Question, i.e. if triggering from a Lead the Template's or Survey's object must be LEAD.

2. **Related Object Id**: Set the Related Object Id to the record ID of the record where the SMS History should show up and even more importantly this governs the Object/Record that an Incoming SMS response will come into.

    a. This field has nothing to do with the Template/Survey object, it only governs where the SMS History appears after it's sent. Note that if using the **Scheduled Time** field to schedule the SMS instead of sending it immediately, this field DOES NOT make it so that the Scheduled SMS shows in the related list of the record. You must explicitly also set that objects ID value as well, e.g. Scheduled_SMS.ContactId

    b. Even though it seems redundant this adds value especially for triggers like Event Reminders where you set the Scheduled SMS Name = Event.Id while setting the Related Object Id = Event.**WhoId** because the Salesforce Event object does not allow related lists like the SMS

History. Other use cases would be triggering from an Opportunity but wanting the SMS to appear in the Contact so that the reply comes back to the Contact record.   However, in this use case you are better off to keep the Related Object Id = Opportunity.Id and use the techniques documented in the section below named "*Roll-up SMS Conversations to Parent Objects*"

3. **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 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.

4. **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, such as Lead.Sticky_Sender__c (more below in the section titled: Dynamic Sender Number. 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 below to strip out all the characters.

```
/*********************************************************
Clean the phone number that is entered to remove all formatting. Handle the
situation where the user adds a + or 1 at the start which otherwise would lead to a
number like 119998887777 and thus an invalid number
*********************************************************/

SUBSTITUTE (
  '+1'
   &
   SUBSTITUTE(
   SUBSTITUTE(
   SUBSTITUTE(
   SUBSTITUTE(
   SUBSTITUTE(
   SUBSTITUTE( Owner.Phone , /*Contact.Owner.Phone*/
   "(",""),
   ")",""),
   " ",""),
   "-",""),
   "+",""),
   ".",""),

   "+11","+1" /*if number was entered as +1(999) 888-7777 we strip out the extra 1 */
)
```

a. We recommend using the special 360SMS custom field named **Sticky_Sender__c** for the Sender Number value. This is a custom formula field that the **Send SMS** buttons and **Conversation View** recognize to override 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.

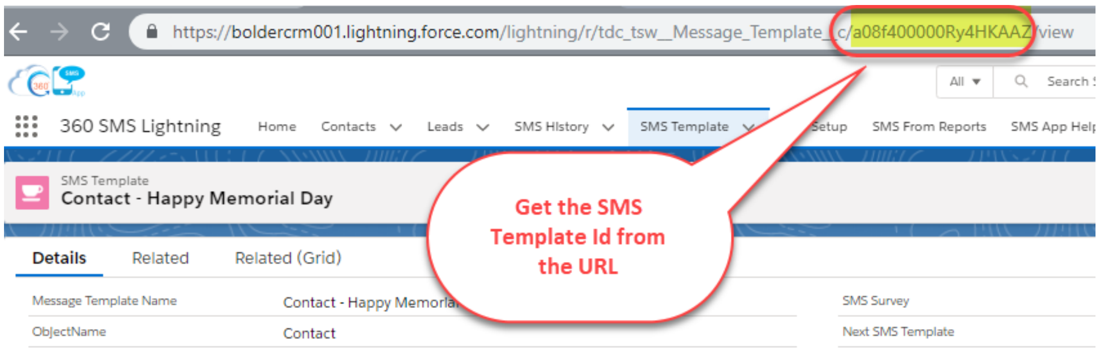
   iii. Area Code matching – many orgs want the SMS Number to match the customers Area Code so it looks like a local number. They use an Area Code formula plus a sticky_sender formula that matches area code to a list of matching SMS Numbers.

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

5. **The Message Text:** There are three ways to set the body of the message that will be sent

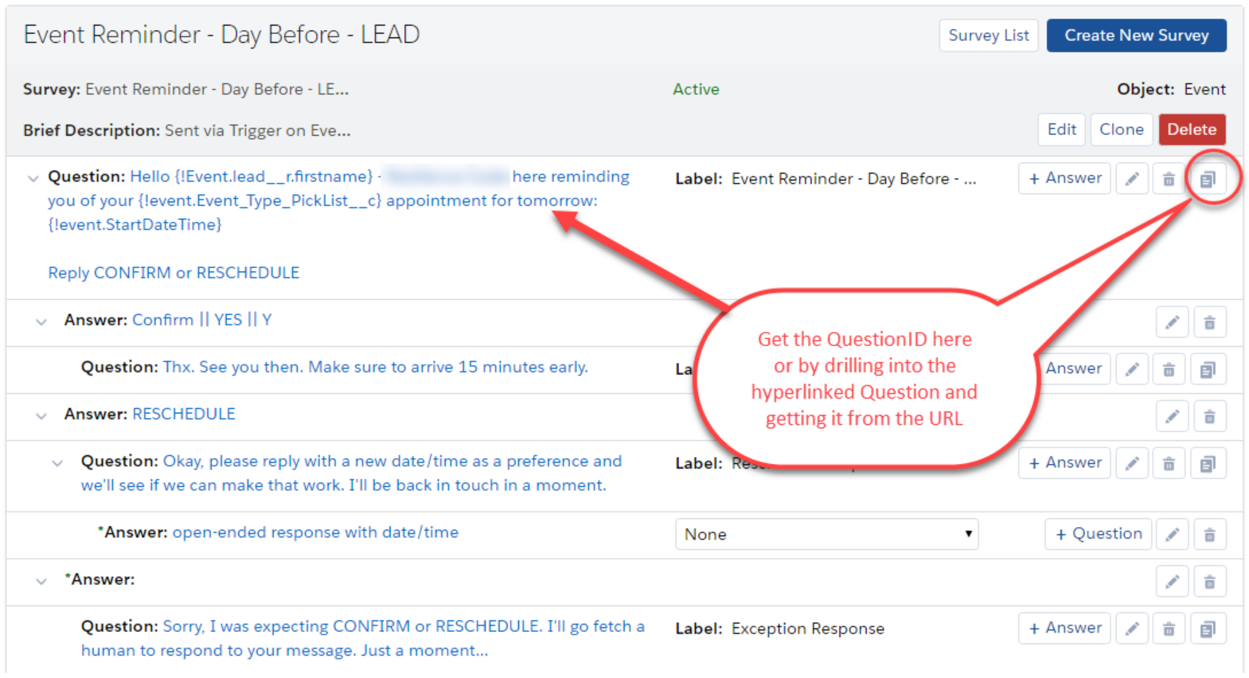
   a. **SMS Template**: Set the ID of the Template to be used. This can be obtained from the URL of the template.

      i. 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*



b. **[iText/Survey] Question**: As an alternative to setting an SMS Template, you may set the **Question** field. The QuestionId is usually the first question of a 360 SMS **Survey**. This then triggers the survey question and responses are automatically triggered from there on. We highly recommend utilizing the Survey object instead of Templates because most outbound SMS will likely generate a response which can be automatically handled by the 360 SMS Survey/ChatBot tool.

i. It is an important, purposeful and brilliant design element that we pass in a Question Id of the survey rather than the Survey Id. Even though 99% of the time you will use the QuestionId of the first question we have seen numerous use cases where it's useful to start a customer in the middle question of some particular survey such as when the survey is serving two purposes and the top of the survey asks questions that a particular customer has already answered or we have the information but we don't want to create a whole new survey so we strategically send them a different question in the survey.


Get the QuestionID here or by drilling into the hyperlinked Question and getting it from the URL

Event Reminder - Day Before - LEAD

Survey List | Create New Survey

Survey: Event Reminder - Day Before - LE... Active **Object:** Event

Brief Description: Sent via Trigger on Eve... Edit | Clone | Delete

Question: Hello {!Event.lead__r.firstname} - [redacted] here reminding you of your {!event.Event_Type_PickList__c} appointment for tomorrow: {!event.StartDateTime}

Reply CONFIRM or RESCHEDULE

Label: Event Reminder - Day Before - ... + Answer

Answer: Confirm || YES || Y

Question: Thx. See you then. Make sure to arrive 15 minutes early. **La** Answer

Answer: RESCHEDULE

Question: Okay, please reply with a new date/time as a preference and we'll see if we can make that work. I'll be back in touch in a moment. **Label:** Res + Answer

*__Answer:__ open-ended response with date/time None + Question

***Answer:**

Question: Sorry, I was expecting CONFIRM or RESCHEDULE. I'll go fetch a human to respond to your message. Just a moment... **Label:** Exception Response + Answer

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

c. **Message Text:** You may create your own message text with a formula or field reference although we rarely recommend this as then you have no way to modify your message text other than modifying your Process Builder. Still it's a powerful and distinct option.

6. **Owner**: As of version 1.196.54 and above you can set the Scheduled_SMS.Owner and the resulting SMS_History.Owner will carry forward. **THIS IS SUPER IMPORTANT** because ownership of the Outgoing message controls who will receive the notification for the incoming reply. See below for a more detailed discussion of ownership implications and techniques.
Controlling SMS History Owner behavior (Incoming Replies Especially) – User Management

7. **Scheduled Time:** The scheduled time field can be left blank in which case it's IMMEDIATELY sent but **WE DO NOT RECOMMEND EVER SENDING IT IMMEDIATELY**, please consider adding at least 1 or 2 minutes to NOW() because it buys you that extra 1 or 2 minutes to delete the Scheduled SMS if you made a mistake and we believe having everything Scheduled creates a more consistent SF User experience and you can troubleshoot 1000% easier. So we say ALWAYS set this field.

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 ID** field is not the same as setting ContactId or LeadId, that field has a different purpose noted below.

b. Usually you will 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 values in Greenwich Mean Time (GMT) which you must compensate for. Below are two common example formulas that compute a specific Date plus Time.

v. Ask about the 360 SMS **Time Zone Anything™** product for dynamic Time Zone and Daylight Savings Time handling.

```
/* Schedule Birthday wish for this yr 16:00 GMT = 8:00am Mountain Time */
/* 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"
               )
```

b. More on Scheduled SMS in these two documents:

i. Triggered Scheduled SMS w/ Process Builder

ii. Triggered Scheduled SMS – Event Reminders

```
/*******************************************************************
SF date math is tricky when wanting a specific time of day such as 8:00am EST. SF wants the
field in format: YYYY-MM-DD HH:MM:SS, plus we have to compensate that SF returns everything in
Greenwich Mean Time so you have to adjust for your time zone.

1. Convert ActivityDateTime (shown in picklist as label TIME) into a text string
2. We just need the YYYY-MM-DD portion of it, so take left 10 characters
3. Now add the time component back adjusting for time zone Eastern Time = GMT minus 5 hrs.
4. Lastly convert it back into a datetimevalue
*******************************************************************/

DATETIMEVALUE(LEFT(TEXT(ActivityDateTime),10) & " 13:00:00")

/* 13:00PM GMT (1:00 PM in UK) = 8:00am EST (GMT -5)*/
```

8. **Attachment/Document Id:** You may set an optional Document.Id value of a picture or other supported document stored in the Salesforce Document object. Refer to the dedicated section below titled: *Send MMS (Pictures or VCards/VCF) via Process* Builder

9. **Channel:** You may specify a channel such as **SMS** or **WhatsApp.** Not using the parameter or setting it to null will send the message as SMS.

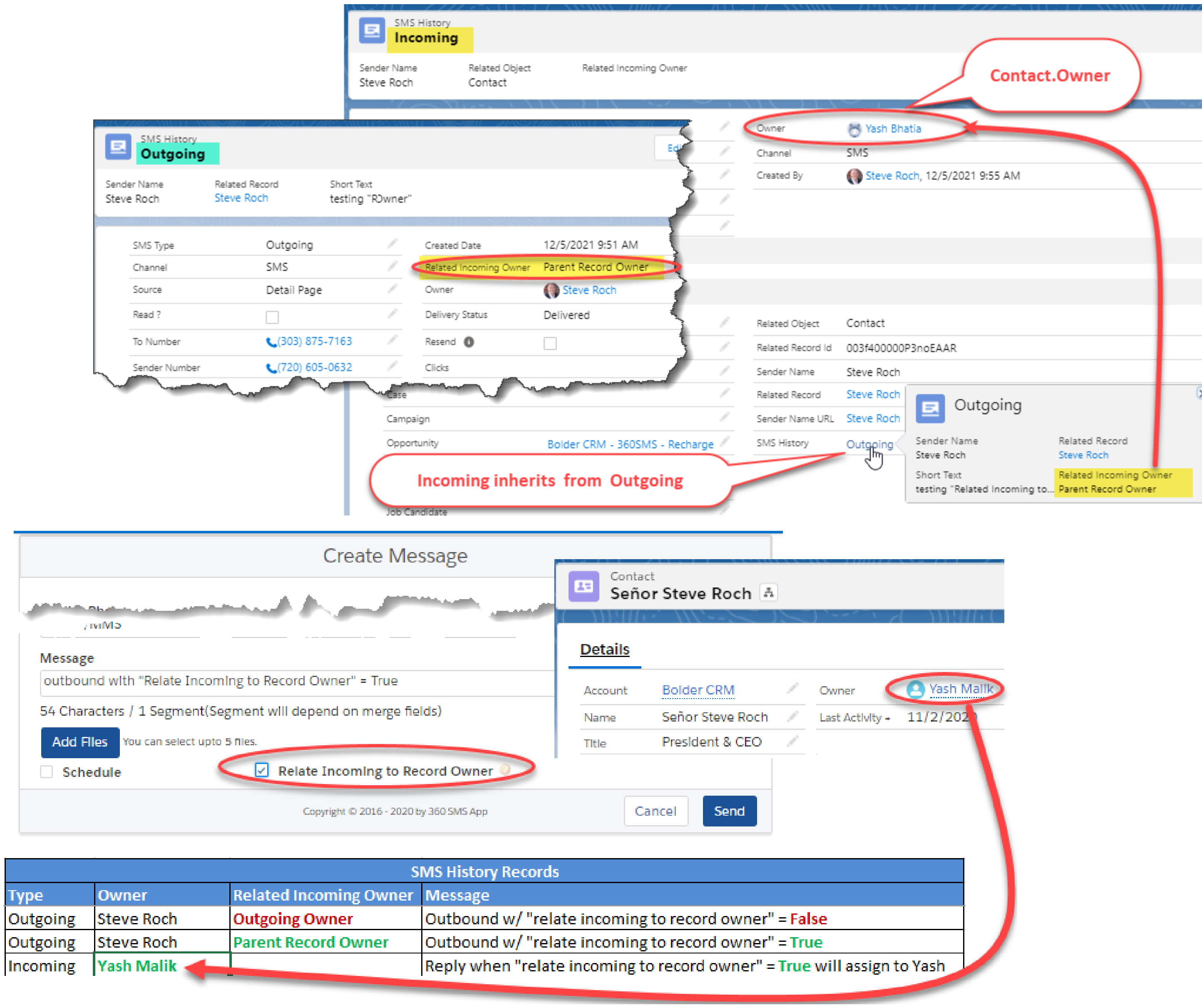
Controlling SMS History Owner behavior (Incoming Replies Especially) – User Management

Whether sending SMS manually or programmatically, a common request is to override the normal behavior for the Incoming SMS Ownership. A key principle of the 360SMS platform is that the ***Incoming inherits from its preceding Outgoing***.

The incoming inherits all lookup field relationships from its preceding Outbound including templates, iText questions and most importantly **OWNER**. And since Owner governs who gets notified of the incoming SMS, developers commonly want to change this behavior.

There are **three methods** to override the platforms natural behavior:

1. **Scheduled_SMS.Owner** - In versions >= **1.196.54** one can now programmatically set Scheduled_SMS.Owner but only with a ScheduledTime and then the resulting SMS_History will indeed set SMS_History.Owner = Scheduled_SMS.Owner. This only works if the ScheduledTime is set such a NOW() + 1/60/24. We recommend all programmatic SMS use a time even if only 1 minute from now because it's just so much easier to debug and creates an opportunity to delete the scheduled SMS. So, this then the whole Ownership discussion is moot because you just set ScheduleSMS.Owner to the desired owner and your set. Just make sure that the Owner + SenderNumber combination is set in SMS User Config or use the "All Users" + SenderNumber technique.
   a. **Important**: This is not a licensing workaround. The current user that is triggering the SMS must still have a 360SMS license. This only affects the outbound SMS_History.Owner value so that the incoming.owner inherits from the outbound.owner.
2. When Sending SMS manually using the "**Send SMS**" button either via Batch or Singularly, the "**Relate Incoming to Record Owner**" feature presents itself when enabled (General Settings). This feature overrides the default inheritance behavior and instead assigns the IncomingSMS.Owner to the SMS_History.Related_Records → Record.Owner. **Figure 3** illustrates how the checkbox sets the field value to "Parent Record Owner" to override the incoming.
   a. Note that the Outgoing.Owner is still the current SF user. The Incoming.Owner will be set to the Contact.Owner when it arrives because of the principle that IN inherits from OUT.
   b. One cannot set the "Relate Incoming to Record Owner" value when triggering an outbound SMS because the Scheduled SMS does not have this field.



SMS History
Incoming

Sender Name	Related Object	Related Incoming Owner
Steve Roch	Contact	

Owner	Yash Bhatia
Channel	SMS
Created By	Steve Roch, 12/5/2021 9:55 AM
Case	
Campaign	
Opportunity	Bolder CRM - 360SMS - Recharge
Job Candidate	
Related Object	Contact
Related Record Id	003f400000P3noEAAR
Sender Name	Steve Roch
Related Record	Steve Roch
Sender Name URL	Steve Roch
SMS History	Outgoing

Outgoing

Sender Name	Related Record
Steve Roch	Steve Roch

Short Text	Related Incoming Owner
testing "Related Incoming to...	Parent Record Owner

SMS History
Outgoing

Sender Name	Related Record	Short Text
Steve Roch	Steve Roch	testing "ROwner"

SMS Type	Outgoing	Created Date	12/5/2021 9:51 AM
Channel	SMS	Related Incoming Owner	Parent Record Owner
Source	Detail Page	Owner	Steve Roch
Read ?	☐	Delivery Status	Delivered
To Number	(303) 875-7163	Resend	☐
Sender Number	(720) 605-0632	Clicks	

Create Message

Message
outbound with "Relate Incoming to Record Owner" = True
54 Characters / 1 Segment(Segment will depend on merge fields)
Add Files You can select upto 5 files.
☐ Schedule ☑ Relate Incoming to Record Owner
Copyright © 2016 - 2020 by 360 SMS App
Cancel Send

Contact
Señor Steve Roch

Details

Account	Bolder CRM	Owner	Yash Malik
Name	Señor Steve Roch	Last Activity	11/2/202
Title	President & CEO		

SMS History Records			
Type	Owner	Related Incoming Owner	Message
Outgoing	Steve Roch	Outgoing Owner	Outbound w/ "relate incoming to record owner" = False
Outgoing	Steve Roch	Parent Record Owner	Outbound w/ "relate incoming to record owner" = True
Incoming	Yash Malik		Reply when "relate incoming to record owner" = True will assign to Yash

Figure 3 – Relate Incoming to Record Owner only sets the Incoming.Owner while retaining the user that sent the outbound

3. Set a Flow/Process Builder on the SMS_History object detecting the SMS outgoing by its TemplateId, or whatever criteria is desired. Then “manually” update the outbound Owner using your business logic. Now, because Incoming.Owner inherits from Outgoing.Owner, problem solved.
    a. Some customers will want other users texting Outbound to customers but they always want the incoming to be assigned to the Contact.Owner.
4. There is an obscure but effective workaround for SMS History owner manipulation via programmatic SMS although with the Scheduled_SMS.Owner this technique is now obsolete:

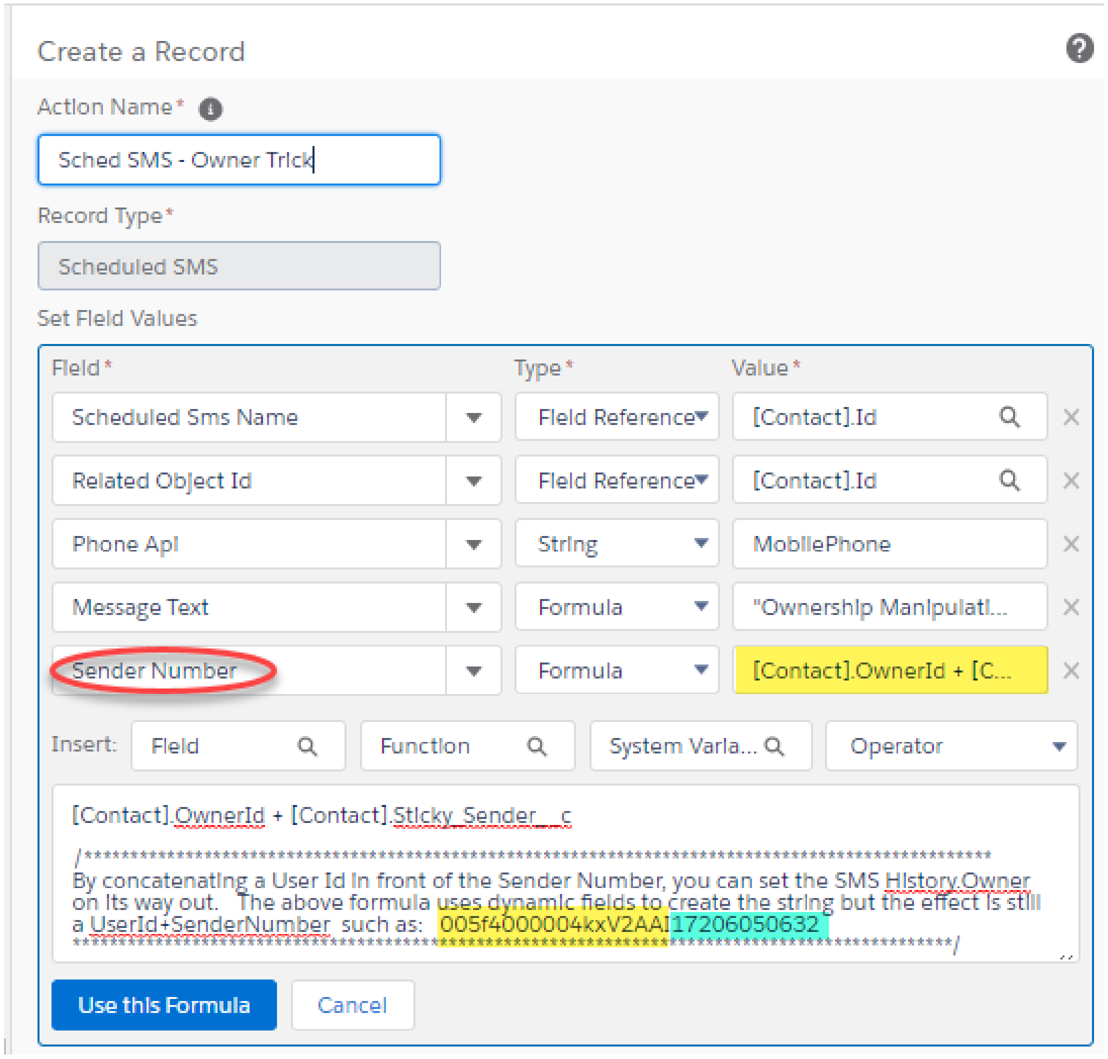
a. Setting the “Sender Number” value to a concatenated string of **UserId** + **Sender_Number** will change the behavior such that the outbound SMS_History.Owner is set to the UserId <u>without</u> any further Process Builder manipulation.

b. Below is a sample process builder setting the outbound SMS.Owner as the Record.Owner

c. **Important**: As of this writing, one cannot set the Scheduled Time value and have this trick work, it only works for immediately triggered SMS.

d. **Important**: This is not a licensing workaround. The current user that is triggering the SMS must still have a 360SMS license. This only affects the SMS_History.Owner value.

e. **Important**: If using the All Users option for User + Number configuration you will need to manufacturer the string with the word “AllUser” + Sender_Number, e.g. AllUser17206050632



Create a Record

Action Name*

Sched SMS - Owner Trick

Record Type*

Scheduled SMS

Set Field Values

Field*	Type*	Value*
Scheduled Sms Name	Field Reference	[Contact].Id
Related Object Id	Field Reference	[Contact].Id
Phone Api	String	MobilePhone
Message Text	Formula	"Ownership Manipulati...
Sender Number	Formula	[Contact].OwnerId + [C...

Insert: Field | Function | System Varia... | Operator

```
[Contact].OwnerId + [Contact].Sticky_Sender__c

/*******************************************************************************************
By concatenating a User Id in front of the Sender Number, you can set the SMS History.Owner
on its way out.   The above formula uses dynamic fields to create the string but the effect is still
a UserId+SenderNumber  such as:   005f4000004kxV2AAI17206050632
*******************************************************************************************/
```

Figure 4 - Set the Sender Number parameter with a concatenated string of UserId+SenderNumber and the resulting outbound SMSHistory.Owner will be the UserId

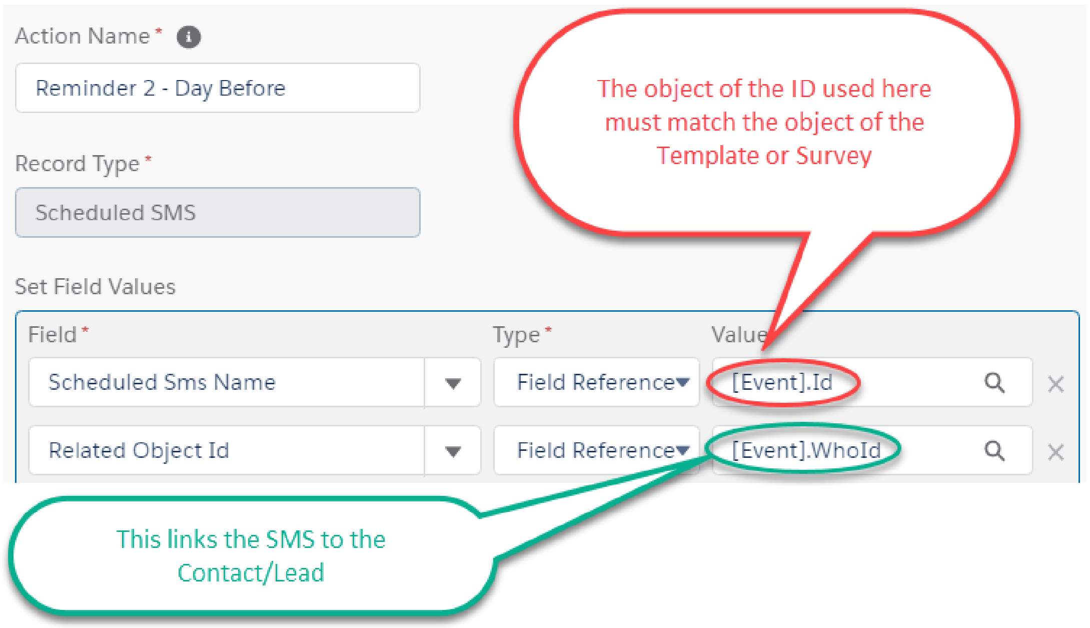
Relating Outbound SMS to an Alternate Object

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. EVENT Reminders and TASK object Triggers
   a. Linking a message to the parent CONTACT/LEAD object when the message is initiated from a Salesforce EVENT or TASK object. This is actually required in this case because the "Black Box" EVENT and TASK objects cannot have any related objects linked to them including SMS History. Jump to section Triggering from Tasks or Events for more info.
2. Linking a message initiated from an Opportunity to a primary Contact or Account
3. 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 msg was sent to the customer.

Below is an example of triggered SMS on the Salesforce Event object where we want the template or survey to be based on the EVENT object so we can merge in the date/time of the event but we want the SMS to appear under the Contact or Lead record.

Read more about the idiosyncrasies of the Event based triggers here: Triggered Scheduled SMS – Event Reminders


The object of the ID used here must match the object of the Template or Survey
This links the SMS to the Contact/Lead

Action Name *
Reminder 2 - Day Before

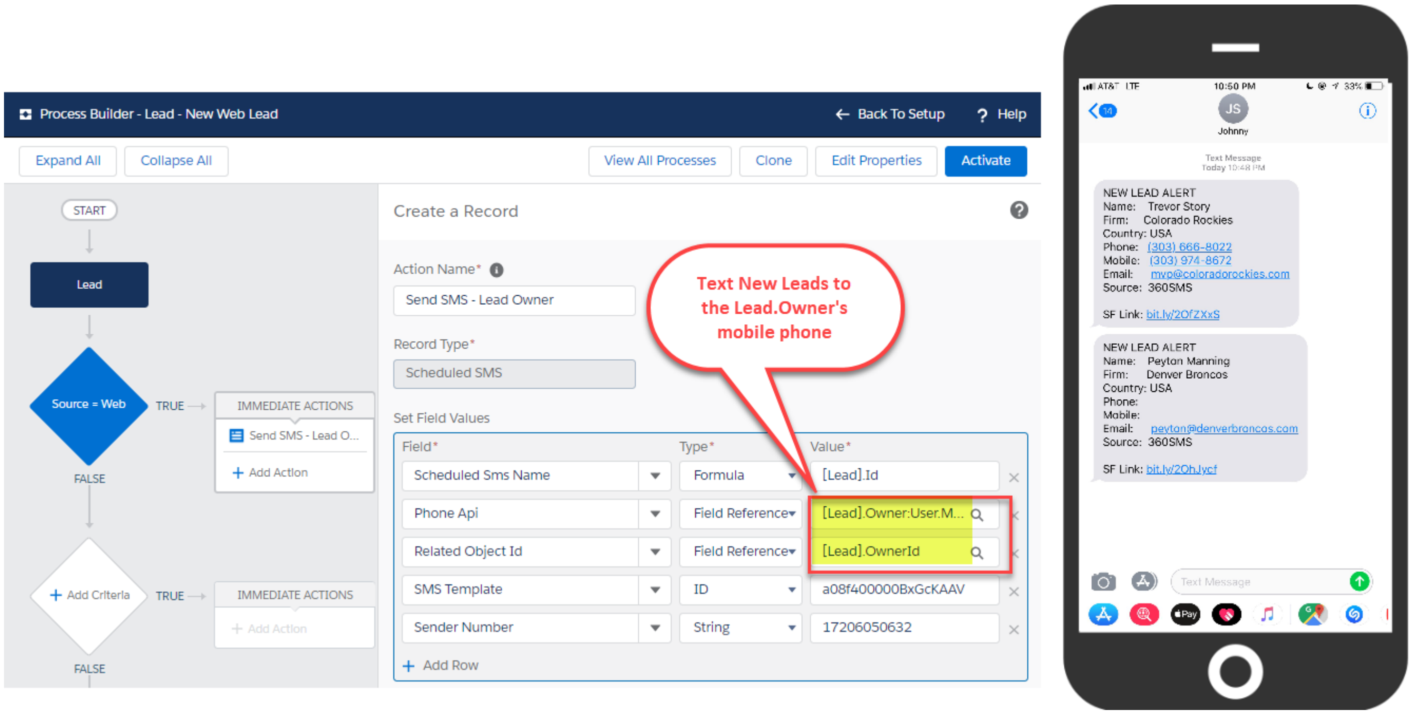
Record Type *
Scheduled SMS

Set Field Values

Field *	Type *	Value
Scheduled Sms Name	Field Reference	[Event].Id
Related Object Id	Field Reference	[Event].WhoId

Figure 5 - Event Reminders should use a Template based on the EVENT but the SMS History will appear under the Contact/Lead

Below is a second 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.



Create a Record

Action Name*: Send SMS - Lead Owner

Record Type*: Scheduled SMS

Set Field Values

Field*	Type*	Value*
Scheduled Sms Name	Formula	[Lead].Id
Phone Api	Field Reference	[Lead].Owner:User.M...
Related Object Id	Field Reference	[Lead].OwnerId
SMS Template	ID	a08f400000BxGcKAAV
Sender Number	String	17206050632

Add Row

AT&T LTE 10:50 PM 33%

Johnny

Text Message
Today 10:48 PM

NEW LEAD ALERT
Name: Trevor Story
Firm: Colorado Rockies
Country: USA
Phone: (303) 666-8022
Mobile: (303) 974-8672
Email: mvp@coloradorockies.com
Source: 360SMS

SF Link: bit.ly/2OfZXxS

NEW LEAD ALERT
Name: Peyton Manning
Firm: Denver Broncos
Country: USA
Phone:
Mobile:
Email: peyton@denverbroncos.com
Source: 360SMS

SF Link: bit.ly/2OhJycf

Figure 6 - Typical internal alert via SMS sending to an employee but triggered from the Lead creation

Read the related section named Roll-up SMS Conversations to Parent Objects to learn a technique where you allow the SMS to be sent/linked from the child object but then with a process builder you additionally set the Ids of the parent objects. This is useful when triggering an SMS from an Opportunity but you want it to appear in the Opportunity as well as the parent Contact and parent Account.

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. There are also Apex classes for Ringless Voicemail and Verify Number.

The Apex class has a single Param field that the Process Builder exposes. You supply a comma separated string into this Param field using formula editor. This method is nice for copying/pasting but we find little use for it beyond that. 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

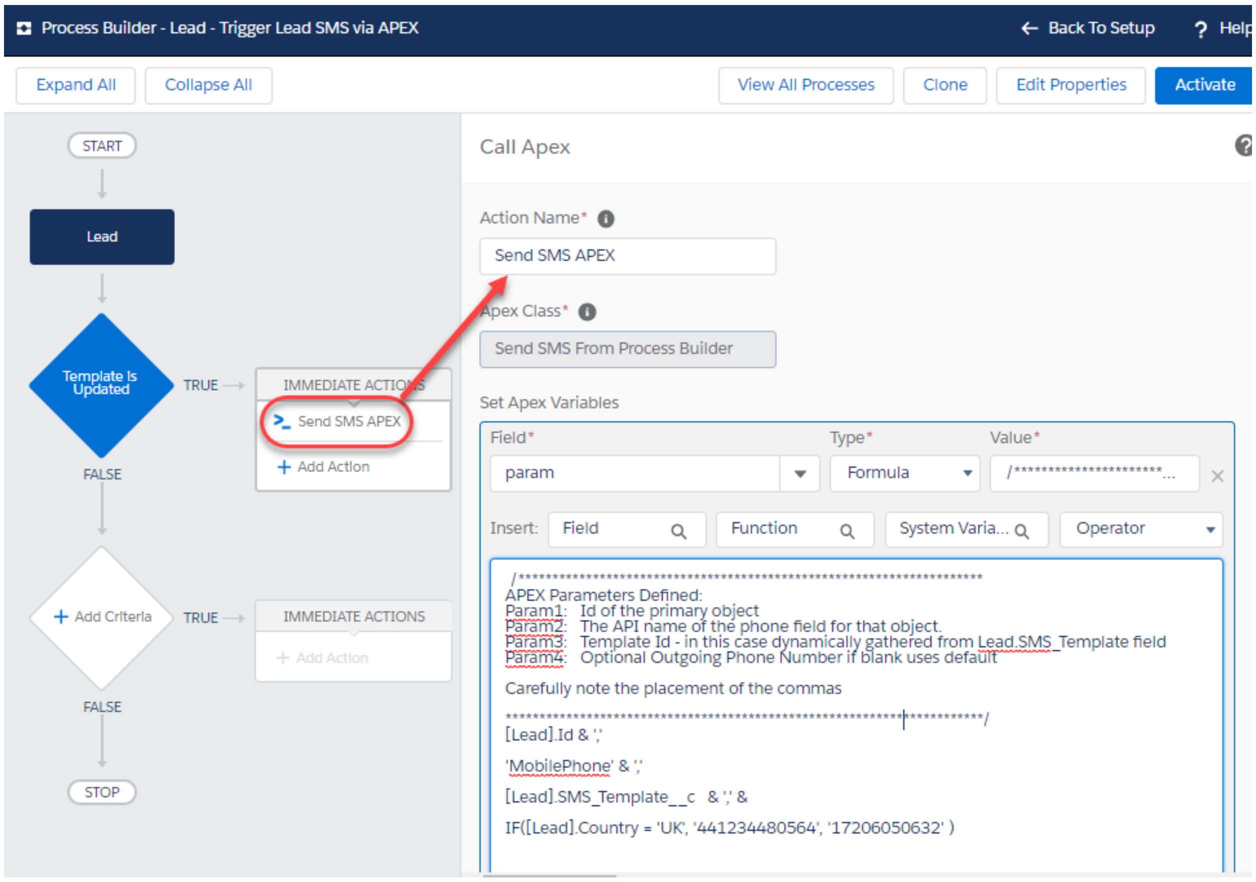
Param3: This parameter can take an SMS TemplateId, a QuestionId or a String.

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.



Action Name*: Send SMS APEX

Apex Class*: Send SMS From Process Builder

Set Apex Variables

Field*	Type*	Value*
param	Formula	/*********************...

Insert: Field | Function | System Varia... | Operator

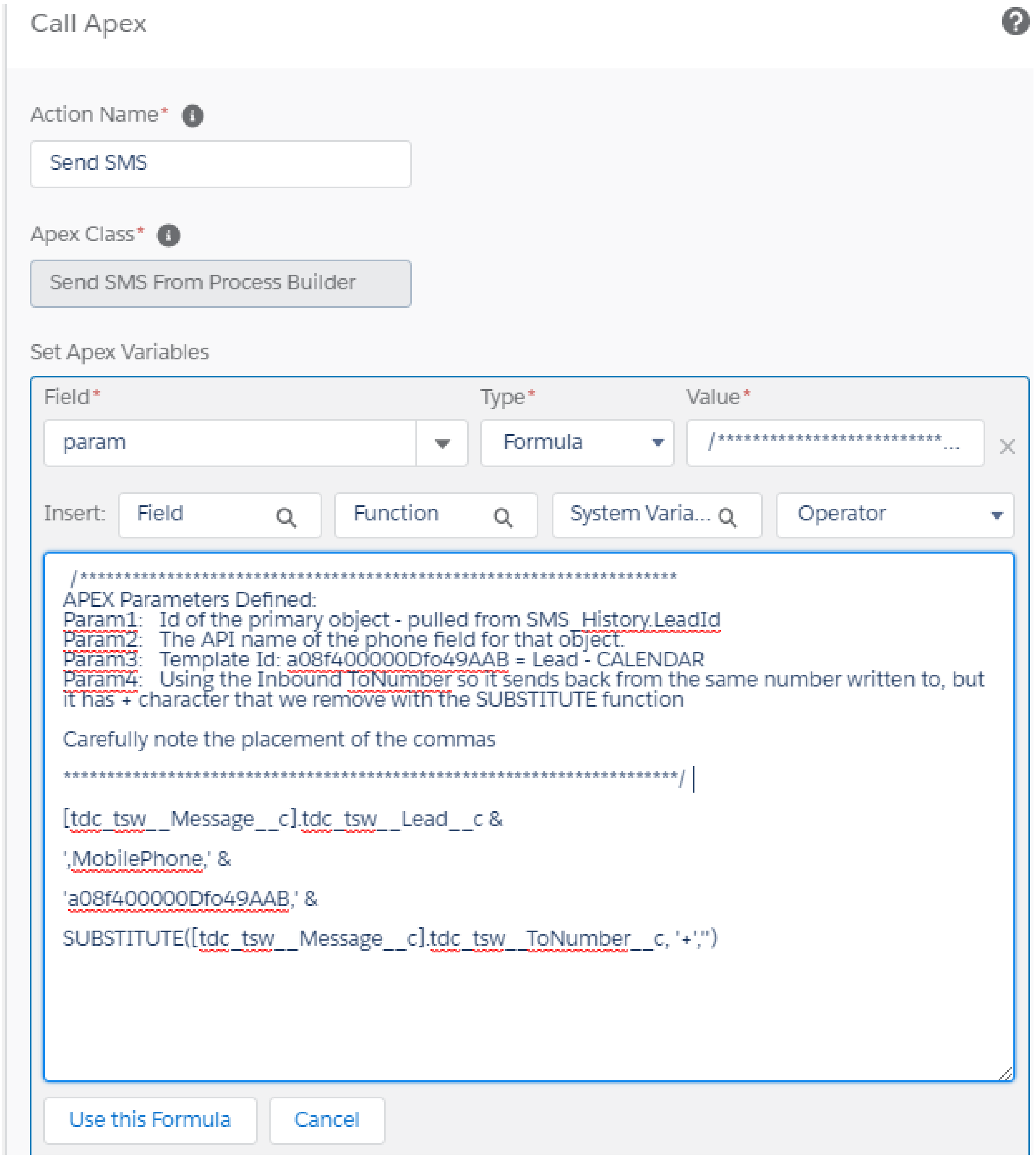
```
/*********************************************************************
APEX Parameters Defined:
Param1:   Id of the primary object
Param2:   The API name of the phone field for that object.
Param3:   Template Id - in this case dynamically gathered from Lead.SMS_Template field
Param4:   Optional Outgoing Phone Number if blank uses default

Carefully note the placement of the commas

*********************************************************************/
[Lead].Id & ','

'MobilePhone' & ','

[Lead].SMS_Template__c   & ',' &

IF([Lead].Country = 'UK', '441234480564', '17206050632' )
```

Figure 7 - 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 ID or referenced field 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. Formula creates a sting like:
003f400000P3noEAAR,MobilePhone,a23f4000000uObmAAE,17206050632
***********************************************************************/
[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 Sender Number

Carefully note the placement of the commas. Formula simply creates a string like:

003f400000P3noEAAR,MobilePhone,a23f4000000uObmAAE,17206050632
***********************************************************************/
[Lead].Id & ',' &
'MobilePhone'  & ',' &
[Lead].SMS_Template__c  & ',' &
[Lead].sticky_sender__c
```


Call Apex

Action Name*

Send SMS

Apex Class*

Send SMS From Process Builder

Set Apex Variables

Field*	Type*	Value*
param	Formula	/**************************...

Insert: Field | Function | System Varia... | Operator

```
/*****************************************************************************
APEX Parameters Defined:
Param1:   Id of the primary object - pulled from SMS_History.LeadId
Param2:   The API name of the phone field for that object.
Param3:   Template Id: a08f400000Dfo49AAB = Lead - CALENDAR
Param4:   Using the Inbound ToNumber so it sends back from the same number written to, but
it has + character that we remove with the SUBSTITUTE function

Carefully note the placement of the commas

*****************************************************************************/

[tdc_tsw__Message__c].tdc_tsw__Lead__c &

',MobilePhone,' &

'a08f400000Dfo49AAB,' &

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

Use this Formula | Cancel

Figure 8 - 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.

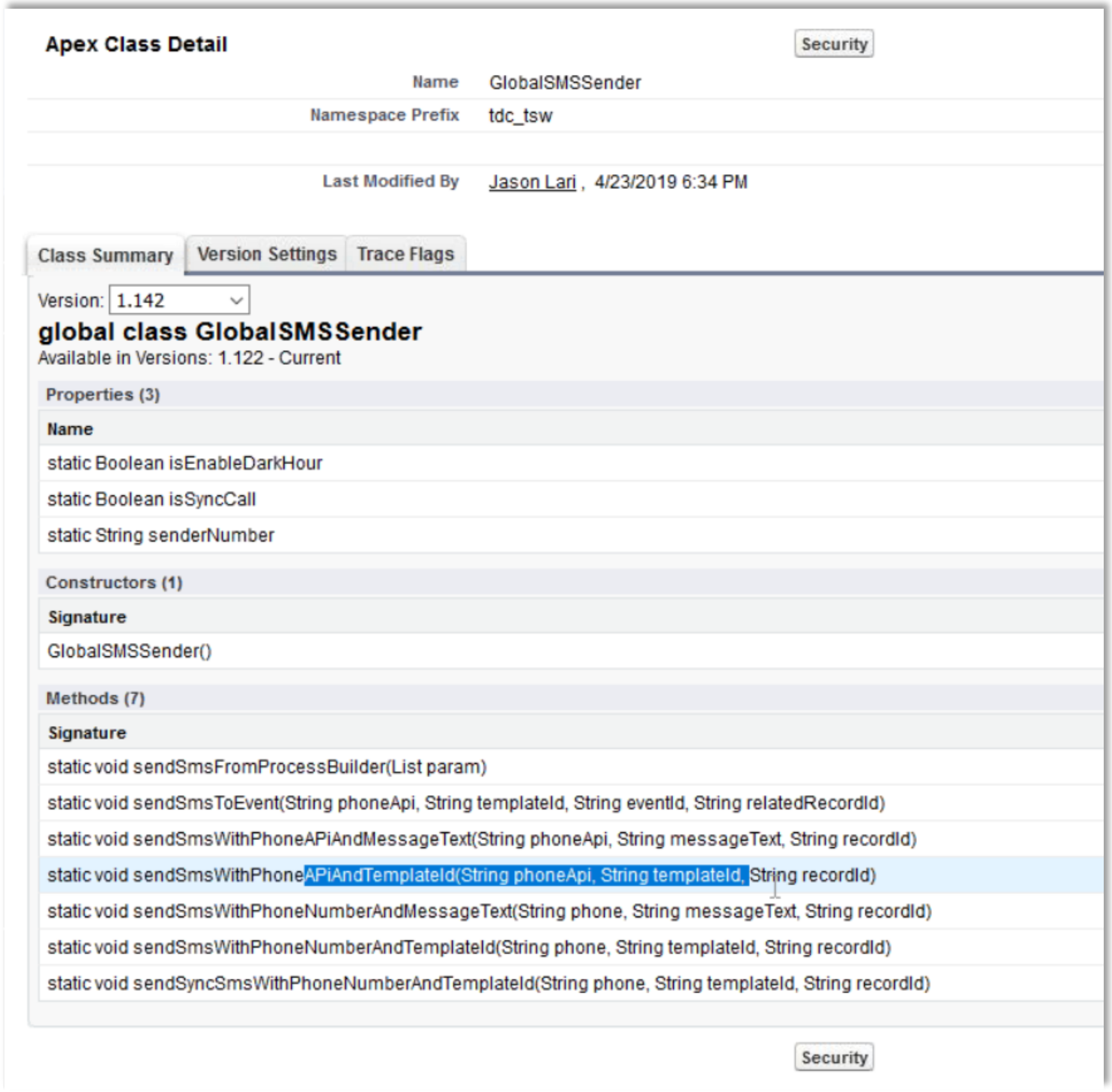
Trigger (for Contact object)

```
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:



Apex Class Detail

Security

Name	GlobalSMSSender
Namespace Prefix	tdc_tsw
Last Modified By	Jason Lari , 4/23/2019 6:34 PM

Class Summary | Version Settings | Trace Flags

Version: 1.142

global class GlobalSMSSender

Available in Versions: 1.122 - Current

Properties (3)

Name
static Boolean isEnableDarkHour
static Boolean isSyncCall
static String senderNumber

Constructors (1)

Signature
GlobalSMSSender()

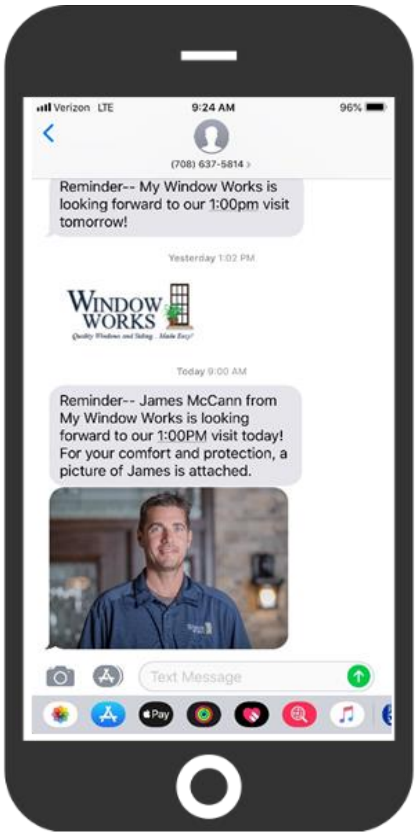
Methods (7)

Signature
static void sendSmsFromProcessBuilder(List param)
static void sendSmsToEvent(String phoneApi, String templateId, String eventId, String relatedRecordId)
static void sendSmsWithPhoneAPiAndMessageText(String phoneApi, String messageText, String recordId)
static void sendSmsWithPhoneAPiAndTemplateId(String phoneApi, String templateId, String recordId)
static void sendSmsWithPhoneNumberAndMessageText(String phone, String messageText, String recordId)
static void sendSmsWithPhoneNumberAndTemplateId(String phone, String templateId, String recordId)
static void sendSyncSmsWithPhoneNumberAndTemplateId(String phone, String templateId, String recordId)

Security

Figure 9 - Additional APEX methods that be called via APEX code

Send MMS (Pictures or VCards/VCF) via Process Builder


Verizon LTE
9:24 AM
96%
(708) 637-5814 ›

Reminder-- My Window Works is looking forward to our 1:00pm visit tomorrow!

Yesterday 1:02 PM

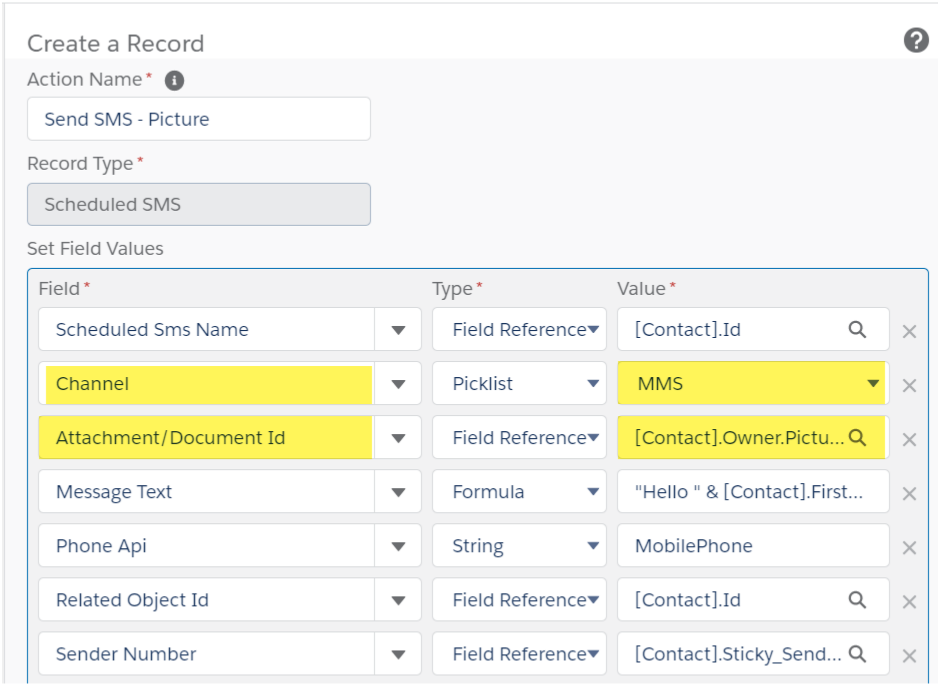
Today 9:00 AM

Reminder-- James McCann from My Window Works is looking forward to our 1:00PM visit today! For your comfort and protection, a picture of James is attached.

MMS is the term for sending or receiving files such as **Pictures** or **VCards (.vcf)** in a text message. A VCard is a special file format that is like sending a Contact record from your phone, such that when the customer clicks it, their phone asks "Do you want to add this as a Contact." Now you are a contact on their phone for incoming Caller Id and SMS.

The key is to set both the Scheduled_SMS.Channel and "Attachment/ Document Id" fields. The "Attachment/ Document Id" must be a previously uploaded Document record whether that be a Picture or a VCard.

Figure 10 below demonstrates a completely dynamic solution where the picture is derived by navigating to a custom field on the USER record such as Contact.Owner.Picture_Doc_Id__c (a custom field holding the Salesforce Document Id of the Salesforce User). Similarly, you could upload a **VCard**(.vcf file) to Salesforce and reference its ID in a custom User field to then trigger an SMS with the users VCard so that customers can quickly add them to their Phone's Contacts by clicking the VCard.


Create a Record

Action Name*: Send SMS - Picture

Record Type*: Scheduled SMS

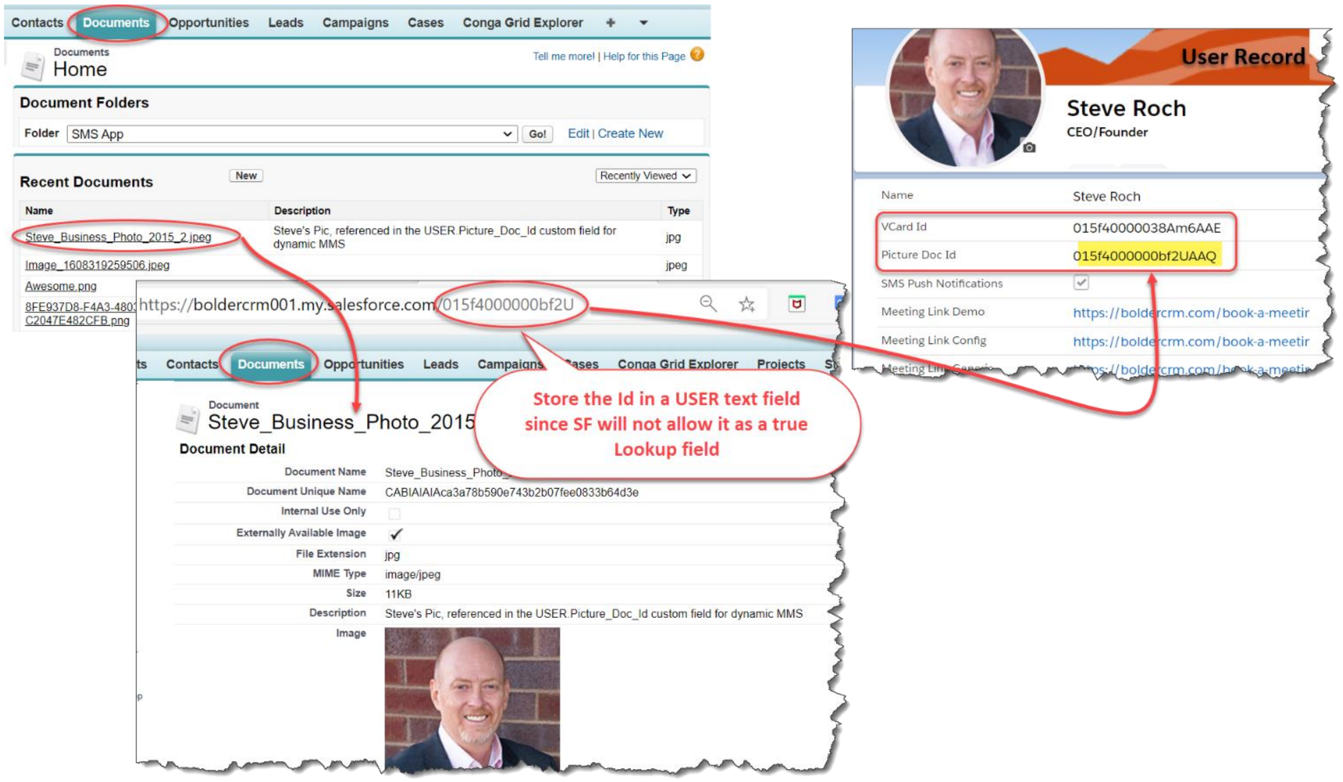
Set Field Values

Field*	Type*	Value*
Scheduled Sms Name	Field Reference	[Contact].Id
Channel	Picklist	MMS
Attachment/Document Id	Field Reference	[Contact].Owner.Pictu...
Message Text	Formula	"Hello " & [Contact].First...
Phone Api	String	MobilePhone
Related Object Id	Field Reference	[Contact].Id
Sender Number	Field Reference	[Contact].Sticky_Send...

Figure 10 - We obtain the picture by traversing up to the USER object and accessing a custom Picture_Doc_Id field.

This dynamic method assumes you have added a Picture or VCard to the native Salesforce **DOCUMENTS** object. This can be a little awkward as Lightning does not expose the DOCUMENTS object anymore, so you must use Classic to upload the files, or you can send yourself an incoming MMS with the picture/vcf file and since 360SMS uses the DOCUMENTS object you can reference it that way.

Screen capture on the next page shows uploading files to the native Salesforce DOCUMENTS object.



Name	Description	Type
Steve_Business_Photo_2015_2.jpeg	Steve's Pic, referenced in the USER.Picture_Doc_Id custom field for dynamic MMS	jpg
Image_1608319259506.jpeg		jpeg
Awesome.png		
8FE937D8-F4A3-480 C2047E482CFB.png		

Document
Steve_Business_Photo_2015

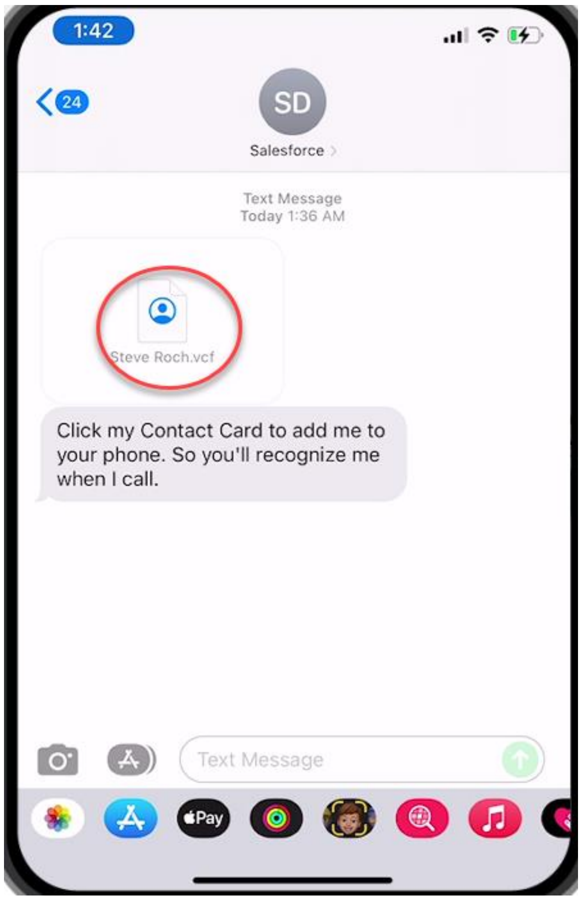
Document Detail

Document Name	Steve_Business_Photo
Document Unique Name	CABIAIAIAca3a78b590e743b2b07fee0833b64d3e
Internal Use Only	
Externally Available Image	✓
File Extension	jpg
MIME Type	image/jpeg
Size	11KB
Description	Steve's Pic, referenced in the USER.Picture_Doc_Id custom field for dynamic MMS
Image	

Name	Steve Roch
VCard Id	015f40000038Am6AAE
Picture Doc Id	015f4000000bf2UAAQ
SMS Push Notifications	✓
Meeting Link Demo	https://boldercrm.com/book-a-meetir
Meeting Link Config	https://boldercrm.com/book-a-meetir

Figure 11 - Uploading Pictures/Files for use in triggered MMS

Sending a .vcf (vCard) is equally simple. First create the VCard in your Email Application of choice and save it somewhere and then upload it to Salesforce like shown in Figure 11. The as shown in Figure 10 simply reference that VCard (Document Id) and set the Channel to MMS.


1:42

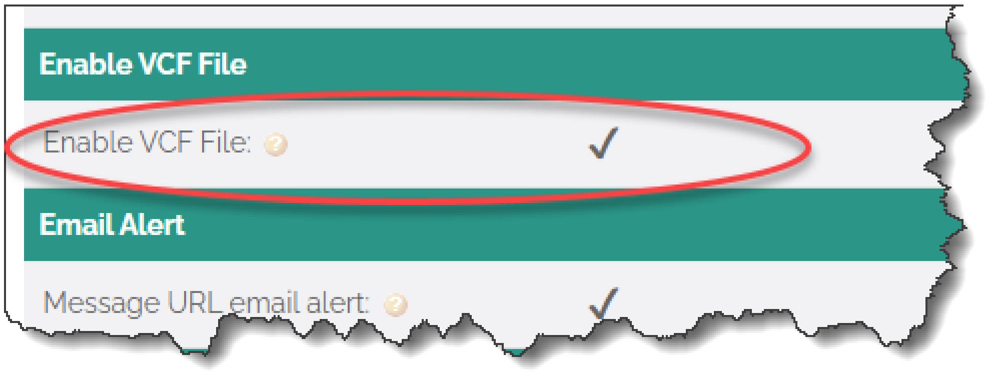
Salesforce

Text Message
Today 1:36 AM

Steve Roch.vcf

Click my Contact Card to add me to your phone. So you'll recognize me when I call.

Important note on VCF: Note that .VCF is a special file type that is slightly different than a regular picture and thus needs to be enabled in the SMS Setup → General Settings as shown below:


Enable VCF File

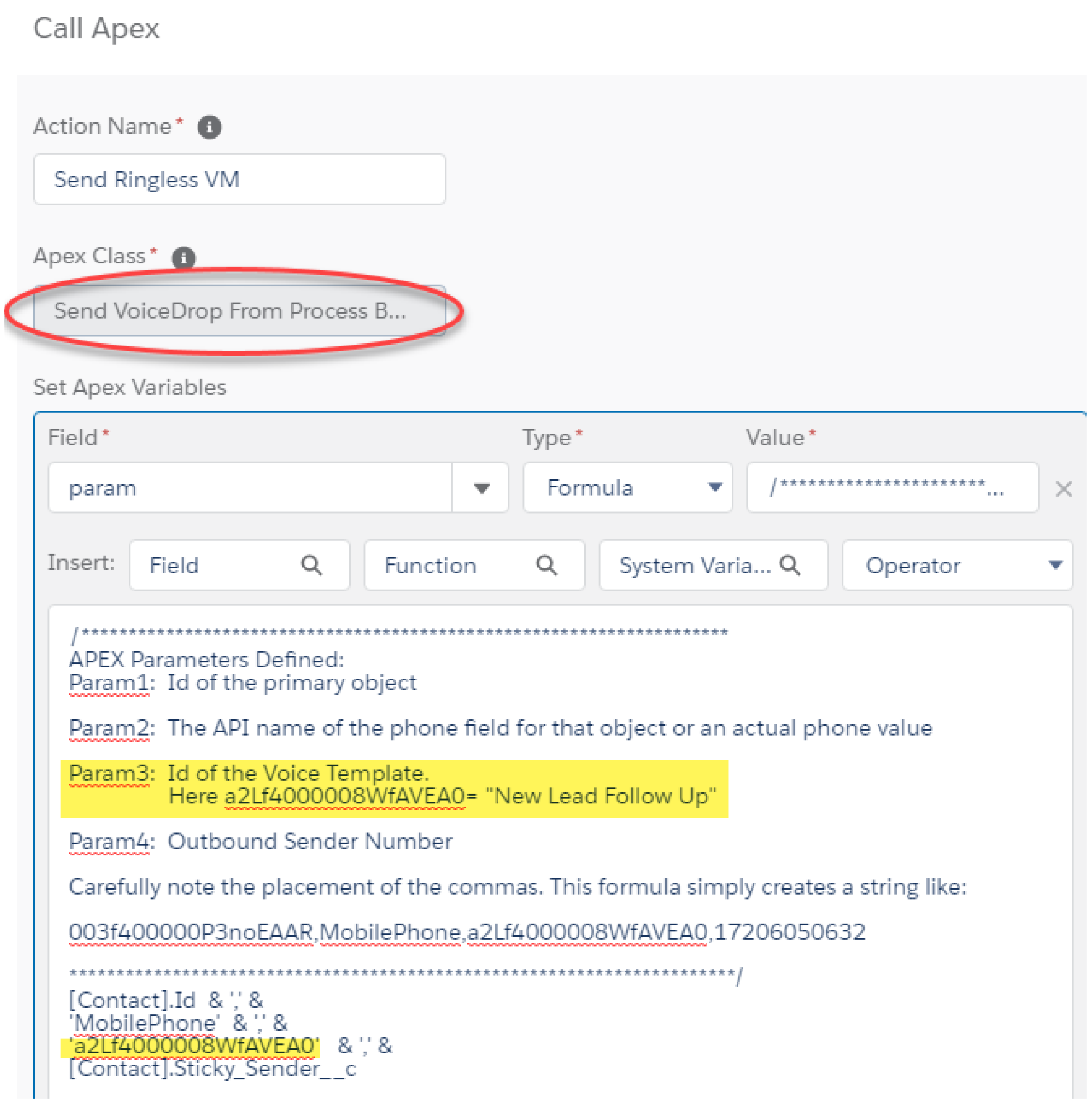
Enable VCF File: ✓

Email Alert

Message URL email alert: ✓

Send Ringless Voicemail via Process Builder

360SMS version **1.154.4** and above offers the Ringless Voicemail Module (purchased separately). A Ringless Voicemail can be triggered just like the above instructions for SMS and MMS using the APEX method, with the only difference being that it uses the APEX class named "**Send VoiceDrop From Process Builder.**" Other than that, it's the same process of constructing a comma separated string to pass into the Param value but in this case, you pass in the ID of the VoiceDrop Template obtained from the record URL just like getting an SMS Template Id.


Call Apex

Action Name* : Send Ringless VM

Apex Class* : Send VoiceDrop From Process B...

Set Apex Variables

Field*	Type*	Value*
param	Formula	/**********************...

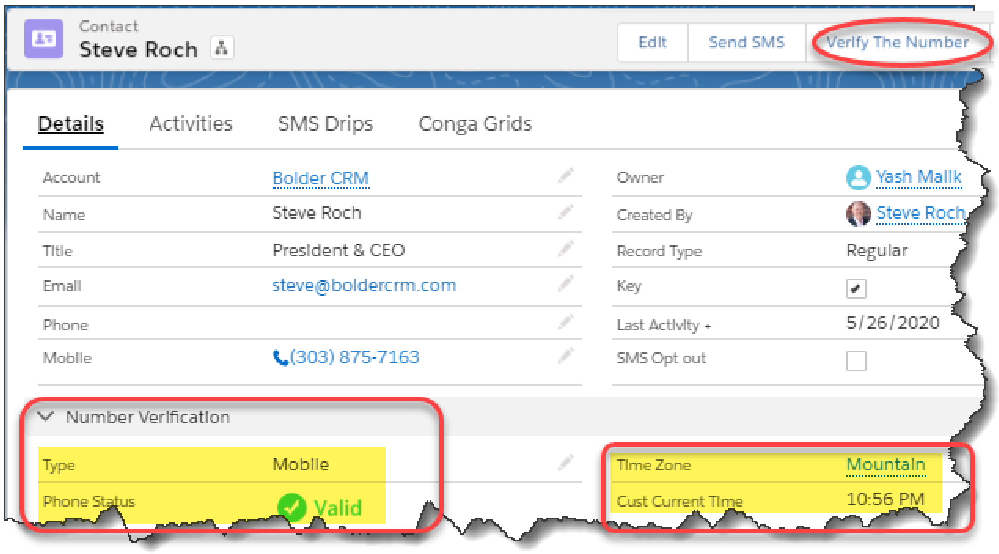
Insert: Field | Function | System Varia... | Operator

```
/*************************************************************************
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:  Id of the Voice Template.
         Here a2Lf4000008WfAVEA0= "New Lead Follow Up"

Param4:  Outbound Sender Number

Carefully note the placement of the commas. This formula simply creates a string like:

003f400000P3noEAAR,MobilePhone,a2Lf4000008WfAVEA0,17206050632

*************************************************************************/
[Contact].Id  & ',' &
'MobilePhone'  & ',' &
'a2Lf4000008WfAVEA0'   & ',' &
[Contact].Sticky_Sender__c
```

Figure 12 - Triggering a Ringless Voicemail is very similar to triggering SMS or MMS

Verify Number via Process Builder

360SMS version **1.162** and above offers the **Verify the Number (VTP)** Module (purchased separately). The feature provides buttons at the Record Detail level and Batch for verifying whether a phone field value is a valid Mobile number or Landline. VTP can be triggered such as when the phone field is changed, or the lead is first created. Similar to triggered SMS, MMS and Ringless Voicemail it uses a simple APEX method, named "**Verify Phone from Process Builder**"   This method is simpler though as it accepts just two parameters, the API name of the Phone field to validate and the RecordId of the record (Contact, Lead, etc.)

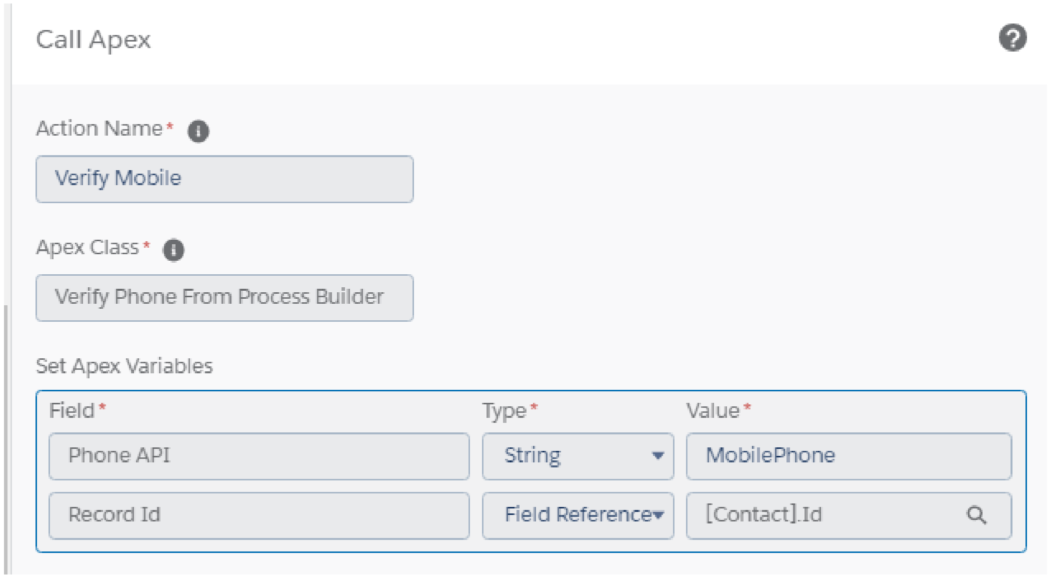


Account	Bolder CRM	Owner	Yash Malik
Name	Steve Roch	Created By	Steve Roch
Title	President & CEO	Record Type	Regular
Email	steve@boldercrm.com	Key	✓
Phone		Last Activity	5/26/2020
Mobile	(303) 875-7163	SMS Opt out	☐

Number Verification

Type	Mobile	Time Zone	Mountain
Phone Status	✓ Valid	Cust Current Time	10:56 PM

*Figure 13 - **Manually** Verify the Number on a single record or in **Batch** from a list view button or **Trigger** it!*


Call Apex

Action Name*: Verify Mobile

Apex Class*: Verify Phone From Process Builder

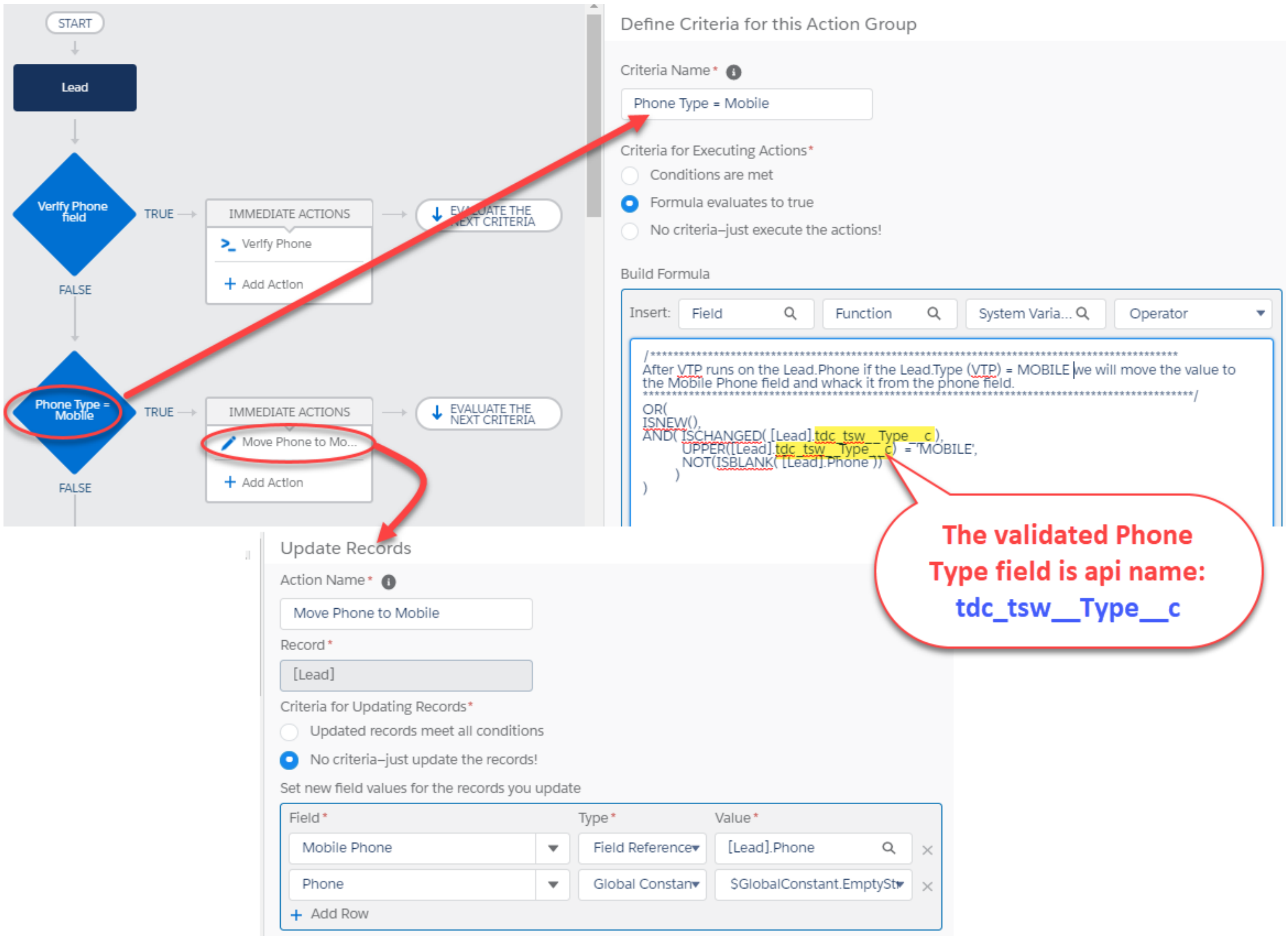
Set Apex Variables

Field*	Type*	Value*
Phone API	String	MobilePhone
Record Id	Field Reference	[Contact].Id

Figure 14 - Verify the Phone can be called with an APEX action in Process Builder

Now you can write a simple Process Builder on the **Lead.Type** or **Contact.Type** to move your phone values into the correct field. The example below assumes that either a website or end-users always enter the phone number into the PHONE field first. Then the VTP process shown in **Figure 14** triggers an update to the Lead.Type field. In Figure 15 we detect that TYPE = "Mobile" and so we know we can move the value from the Lead.Phone to the Lead.MobilePhone.

This trigger also works fabulously when using the VTP button on a List View (Batch VTP). Simply add this trigger first and then batch verify all your Phone fields and it will "automatically" move the values into the MobilePhone field for your entire database or leave them if the number is validated as a Landline. **Tremendous value!!**



Define Criteria for this Action Group

Criteria Name*: Phone Type = Mobile

Criteria for Executing Actions*
- Conditions are met
- [✓ Formula evaluates to true]
- No criteria—just execute the actions!

Build Formula

Insert: Field | Function | System Varia... | Operator

```
/*************************************************************************
After VTP runs on the Lead.Phone if the Lead.Type (VTP) = MOBILE we will move the value to
the Mobile Phone field and whack it from the phone field.
***************************************************************************/
OR(
ISNEW(),
AND( ISCHANGED( [Lead].tdc_tsw__Type__c ),
      UPPER([Lead].tdc_tsw__Type__c) = 'MOBILE',
      NOT(ISBLANK( [Lead].Phone ))
    )
)
```

Update Records

Action Name*: Move Phone to Mobile

Record*: [Lead]

Criteria for Updating Records*
- Updated records meet all conditions
- [✓ No criteria—just update the records!]

Set new field values for the records you update

Field*	Type*	Value*
Mobile Phone	Field Reference	[Lead].Phone
Phone	Global Constant	$GlobalConstant.EmptySt

Add Row

Figure 15 - After verifying a number - trigger the movement of the value to the correct field (Phone --> MobilePhone in this case)

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 16 - Useful formula for reducing a number to 17206050632 format

There are four common use cases:

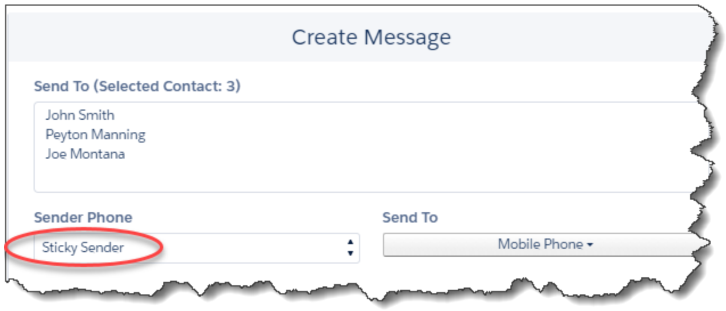
1. **Record Based Sender Number** – assign a specific sender number based on business logic such that all SMS to that Contact or Lead always comes from a single sender number whether that be Record.Owner based, Geography based or defined by other business logic.
2. **Dynamically set the Sender Number 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 iText/Survey tool does this automatically – it always responds with 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, Area Code matching numbers, Marketing Campaign Based Numbers, 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.

Read the document Area Code Matching SMS Numbers for a specific detailed example.

If the Sticky_Sender field exists, all interface elements of the platform (Buttons, Conversation View and SMS From Reports) will override the current users Default SMS Number and instead offer this field value 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 pull from the sticky sender as its looping through the records. This also helps distribute batch SMS among multiple numbers thereby reducing spam detection.

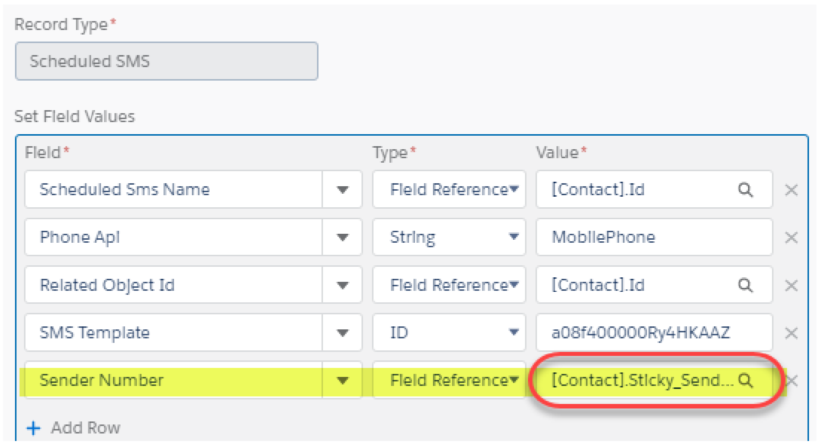

Create Message

Send To (Selected Contact: 3)
John Smith
Peyton Manning
Joe Montana

Sender Phone: Sticky Sender
Send To: Mobile Phone

Figure 17 - 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.


Record Type*: Scheduled SMS

Set Field Values

Field*	Type*	Value*
Scheduled Sms Name	Field Reference	[Contact].Id
Phone Api	String	MobilePhone
Related Object Id	Field Reference	[Contact].Id
SMS Template	ID	a08f400000Ry4HKAAZ
Sender Number	Field Reference	[Contact].Sticky_Send...

+ Add Row

Figure 18 - 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). Careful, cuz SF will
display a phone field in format (303) 875-7163 even if the
underlying data is entered at 1303-875-7163 so this formula
handles that.
***************************************************************/
SUBSTITUTE (
  '+1'
   &
   SUBSTITUTE(
   SUBSTITUTE(
   SUBSTITUTE(
   SUBSTITUTE(
   SUBSTITUTE(
   SUBSTITUTE( Owner.Phone ,
   "(",""),
   ")",""),
   " ",""),
   "-",""),
   "+",""),
   ".",""),

   "+11","+1" /*if the # was entered like 19998887777 we strip out the extra 1 */
)
```

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 , '+', '')
```

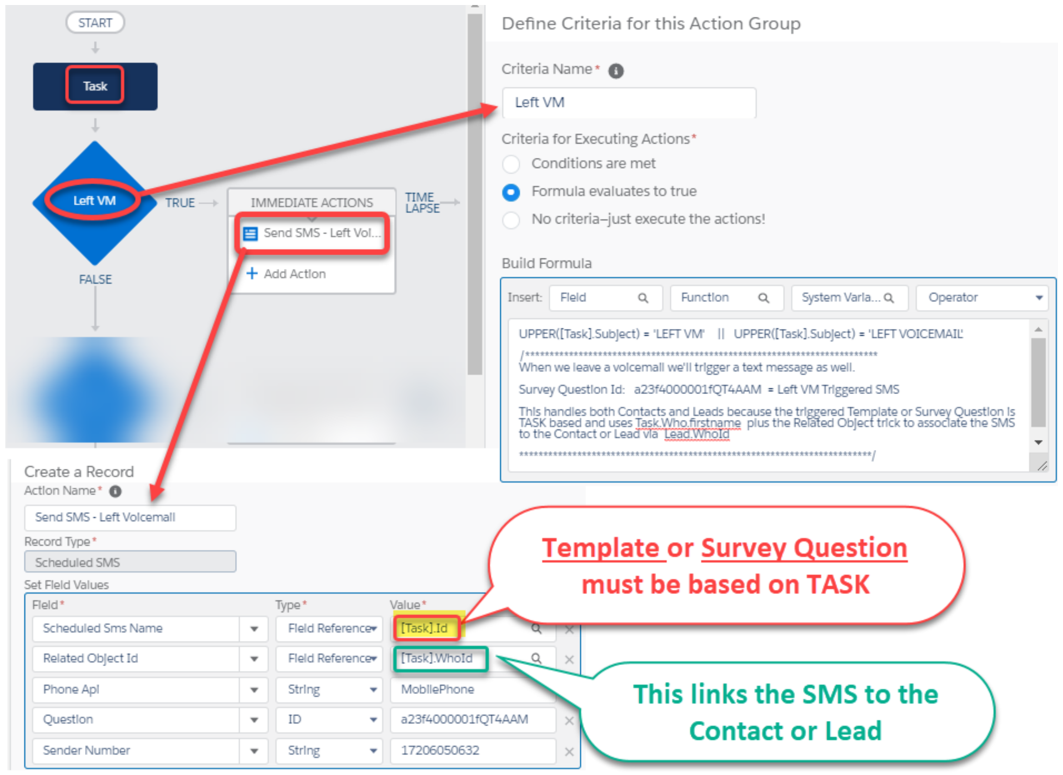
Triggering from Tasks or Events

The Salesforce **Task** and **Event** objects are common objects to trigger off of but require special understanding and handling because of the challenging design of these native Salesforce objects. With that in mind the dedicated document named SMS Event Reminders discusses the challenges and methods at length. Here we will simply state that the challenge is that the TASK and EVENT objects use the **NAME** and **RELATED TO** fields (api names: WhoId and WhatId respectively) as polymorphic ID fields, meaning that these special fields can hold either a LeadId or a ContactId and the Related To can hold practically any object Id.

This causes Salesforce developers around the world great frustration because neither the Salesforce Formula Editor nor Process Builder can traverse to the Lead or Contact tables because Salesforce doesn't know which object to go to, so it's completely blocked.

Furthermore, neither the EVENT nor the TASK object can have a child related list. This impacts 360SMS (and all other SMS apps) because it's not a normal object that can have SMS related to it.

Luckily 360SMS has been designed with this in mind. Above you learned about the value of the **Related Object Id** parameter. Here is the exact reason for that feature, because usually you'll want merge tag information from the Task/Event especially (Date, Time, etc.) but because you have no phone info on the Task/Event and you want the SMS to appear under the Contact/Lead you'll set the RelatedObject to the Contact or Lead. **Brilliant**!


START
Task
Left VM
TRUE
FALSE
IMMEDIATE ACTIONS
Send SMS - Left Vol...
Add Action
TIME LAPSE
Template or Survey Question must be based on TASK
This links the SMS to the Contact or Lead

Define Criteria for this Action Group

Criteria Name*: Left VM

Criteria for Executing Actions*
- Conditions are met
- [✓ Formula evaluates to true]
- No criteria–just execute the actions!

Build Formula

Insert: Field | Function | System Varia... | Operator

```
UPPER([Task].Subject) = 'LEFT VM'   ||  UPPER([Task].Subject) = 'LEFT VOICEMAIL'
/****************************************************************************
When we leave a voicemail we'll trigger a text message as well.

Survey Question Id:   a23f4000001fQT4AAM  = Left VM Triggered SMS

This handles both Contacts and Leads because the triggered Template or Survey Question is
TASK based and uses Task.Who.firstname  plus the Related Object trick to associate the SMS
to the Contact or Lead via  Lead.WhoId

****************************************************************************/
```

Create a Record

Action Name*: Send SMS - Left Voicemail

Record Type*: Scheduled SMS

Set Field Values

Field*	Type*	Value*
Scheduled Sms Name	Field Reference	[Task].Id
Related Object Id	Field Reference	[Task].WhoId
Phone Api	String	MobilePhone
Question	ID	a23f4000001fQT4AAM
Sender Number	String	17206050632

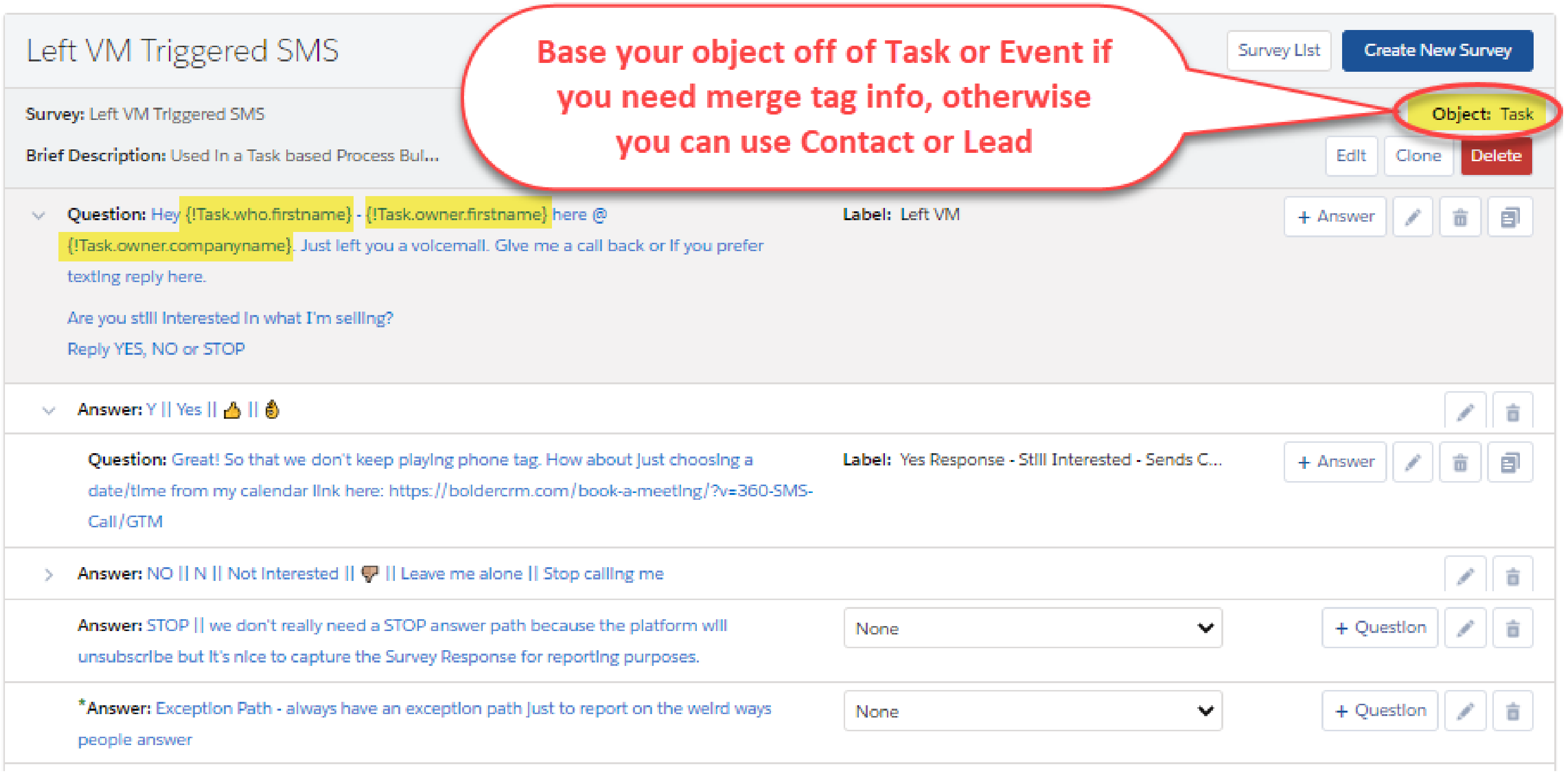
In the example above we are triggering a simple SMS from the Task object when a user sets the Subject to “Left Voicemail”.

Most importantly, we set the Related Object Id = Task.**WhoId** (contact or Lead). If we don’t need any merge tag data from the Task then we could have simply set the “Scheduled SMS Name” field = Task.WhoId as well and used a CONTACT/LEAD based template but then we would need two Process Builder Criteria nodes to detect that the WhoId is either a Contact Id `LEFT([Task.WhoId],3) = '003'` or LeadId `LEFT ([Task.WhoId],3) = '00Q'` .

BTW: Every SF org in the world uses the same object Id prefixes for Contacts (003) or Leads (00Q) and this is the ONLY way to detect which object a task/event is linked to. Wouldn’t you think Salesforce would give us separate fields or some easier way to detect the differences. It’s been 15 years already!

With the EVENT object it’s very common to want to merge in the Date, Time and other values so one almost always bases their Template or Survey Question off of the EVENT object. With the TASK object it’s probably not necessary to use a Task based template but I was lazy and didn’t want to create two separate Process Builder criteria’s and Immediate Actions plus two separate Lead and Contact based Templates or Survey Questions. So this Survey Question below handles either Leads or Contacts.

Note that Salesforce does offer a very limited number of fields from the WHOID relationship via formulas that they have created but only FirstName and LastName not much else. As noted above Salesforce does not let you create formulas traversing from the WhoId to the Contact/Lead!


Survey List
Create New Survey
Base your object off of Task or Event if you need merge tag info, otherwise you can use Contact or Lead
Edit
Clone
Delete
+ Answer
+ Question

Left VM Triggered SMS

Survey: Left VM Triggered SMS

Object: Task

Brief Description: Used In a Task based Process Bul...

Question: Hey {!Task.who.firstname} - {!Task.owner.firstname} here @ {!Task.owner.companyname}. Just left you a voicemail. Give me a call back or if you prefer texting reply here.

Are you still interested in what I'm selling?
Reply YES, NO or STOP

Label: Left VM

Answer: Y || Yes || 👍 || 👌

Question: Great! So that we don't keep playing phone tag. How about just choosing a date/time from my calendar link here: https://boldercrm.com/book-a-meeting/?v=360-SMS-Call/GTM

Label: Yes Response - Still Interested - Sends C...

Answer: NO || N || Not Interested || 👎 || Leave me alone || Stop calling me

Answer: STOP || we don't really need a STOP answer path because the platform will unsubscribe but it's nice to capture the Survey Response for reporting purposes.

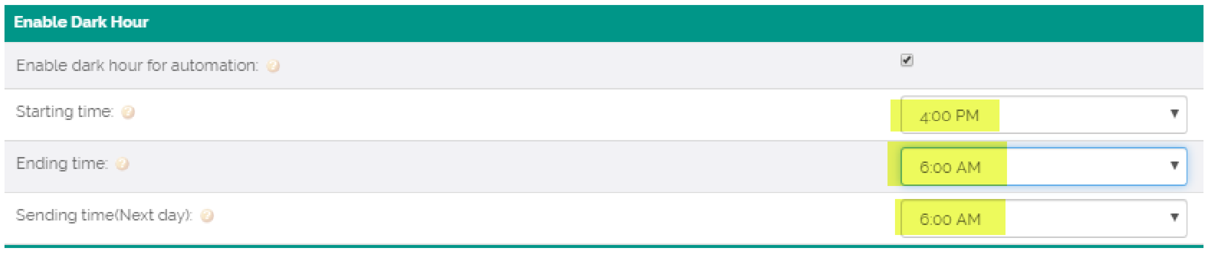
None

*Answer: Exception Path - always have an exception path just to report on the weird ways people answer

None

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.

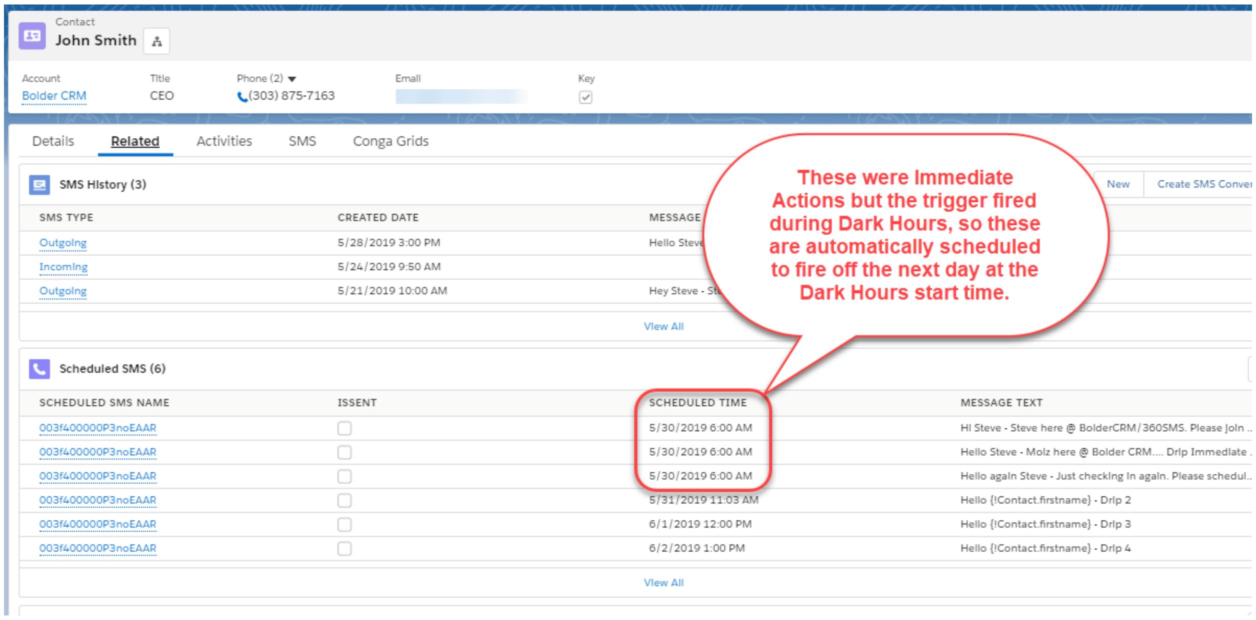


Enable Dark Hour

Enable dark hour for automation:	✓
Starting time:	4:00 PM
Ending time:	6:00 AM
Sending time(Next day):	6:00 AM

Figure 19 - 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.



Contact
John Smith

Account	Title	Phone (2)	Email	Key
Bolder CRM	CEO	(303) 875-7163		✓

SMS History (3)

SMS TYPE	CREATED DATE	MESSAGE
Outgoing	5/28/2019 3:00 PM	Hello Steve
Incoming	5/24/2019 9:50 AM	
Outgoing	5/21/2019 10:00 AM	Hey Steve - St

These were Immediate Actions but the trigger fired during Dark Hours, so these are automatically scheduled to fire off the next day at the Dark Hours start time.

Scheduled SMS (6)

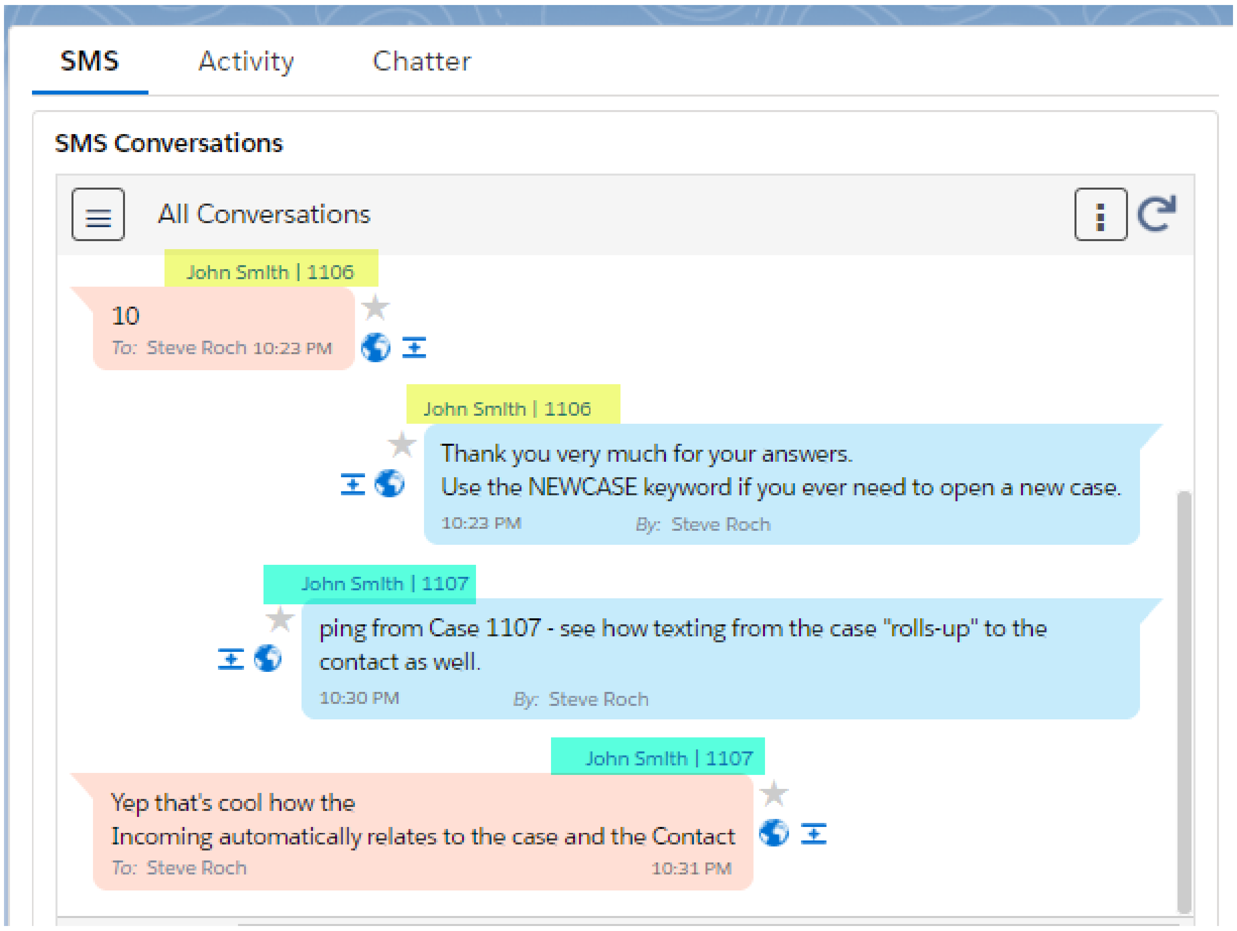
SCHEDULED SMS NAME	ISSENT	SCHEDULED TIME	MESSAGE TEXT
003f400000P3noEAAR	☐	5/30/2019 6:00 AM	Hi Steve - Steve here @ BolderCRM/360SMS. Please join ..
003f400000P3noEAAR	☐	5/30/2019 6:00 AM	Hello Steve - Molz here @ Bolder CRM.... Drip Immediate .
003f400000P3noEAAR	☐	5/30/2019 6:00 AM	Hello again Steve - Just checking in again. Please schedul..
003f400000P3noEAAR	☐	5/31/2019 11:03 AM	Hello {!Contact.firstname} - Drip 2
003f400000P3noEAAR	☐	6/1/2019 12:00 PM	Hello {!Contact.firstname} - Drip 3
003f400000P3noEAAR	☐	6/2/2019 1:00 PM	Hello {!Contact.firstname} - Drip 4

Figure 20 - 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 is 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 a Contact it should roll-up to the Account so from the Account page one can see all the conversations with all the contacts.

Straight out-of-the-box the CASE object automatically rolls-up any SMS linked to a 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.



John Smith | 1106

10
To: Steve Roch 10:23 PM

John Smith | 1106

Thank you very much for your answers.
Use the NEWCASE keyword if you ever need to open a new case.
10:23 PM By: Steve Roch

John Smith | 1107

ping from Case 1107 - see how texting from the case "rolls-up" to the contact as well.
10:30 PM By: Steve Roch

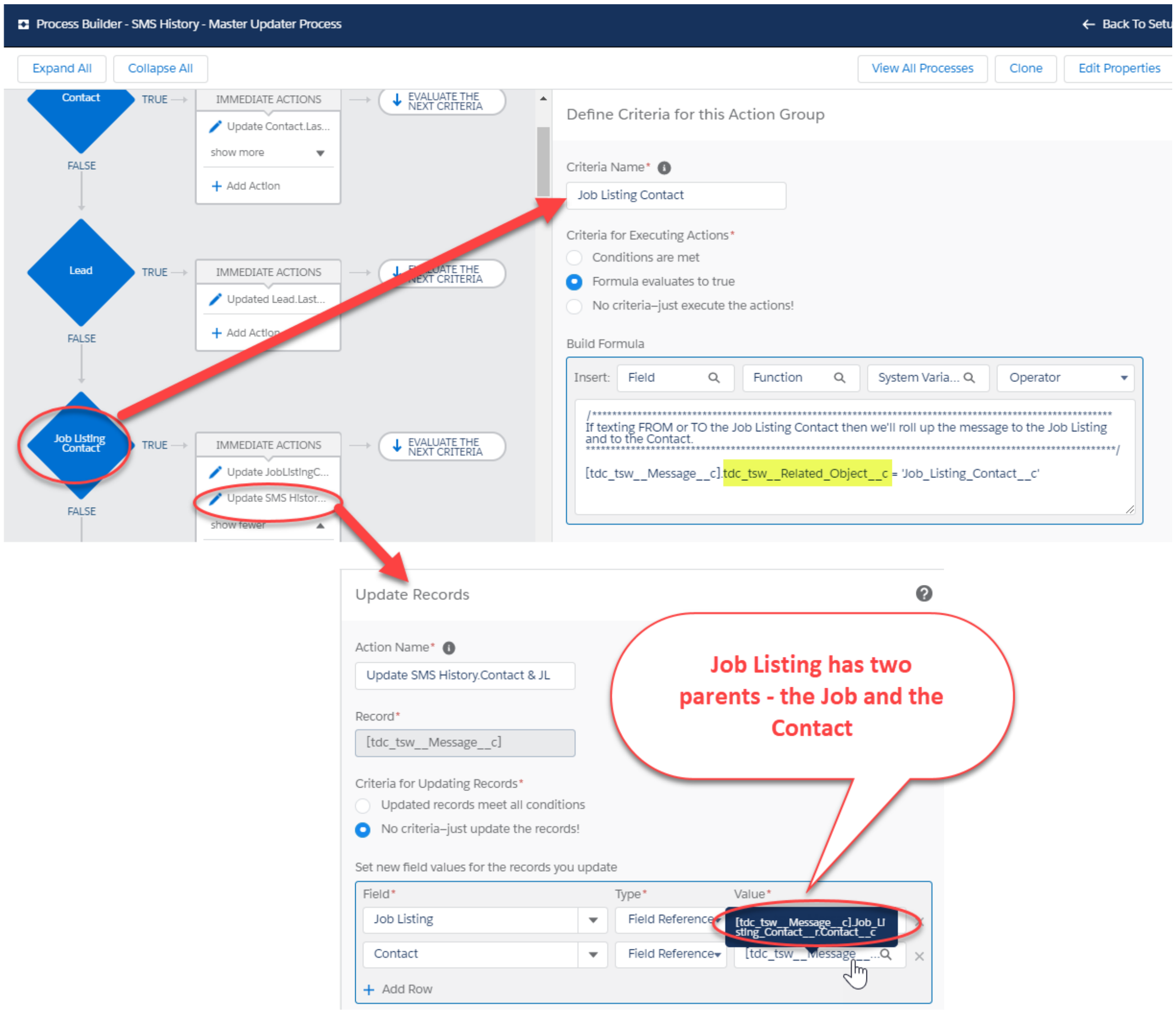
John Smith | 1107

Yep that's cool how the
Incoming automatically relates to the case and the Contact
To: Steve Roch 10:31 PM

*Figure 21 - 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 22*), we have a classic child object named Job_Listing which is the child of a Job and 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 21* 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`



Define Criteria for this Action Group

Criteria Name*: Job Listing Contact

Criteria for Executing Actions*
Conditions are met
[✓ Formula evaluates to true]
No criteria–just execute the actions!

Build Formula

```
/*******************************************************************************************
If texting FROM or TO the Job Listing Contact then we'll roll up the message to the Job Listing
and to the Contact.
*******************************************************************************************/

[tdc_tsw__Message__c].tdc_tsw__Related_Object__c = 'Job_Listing_Contact__c'
```

Update Records

Action Name*: Update SMS History.Contact & JL

Record*: [tdc_tsw__Message__c]

Criteria for Updating Records*
Updated records meet all conditions
[✓ No criteria–just update the records!]

Set new field values for the records you update

Field*	Type*	Value*
Job Listing	Field Reference	[tdc_tsw__Message__c].Job_Listing_Contact__r.Contact__c
Contact	Field Reference	[tdc_tsw__Message__...

Add Row

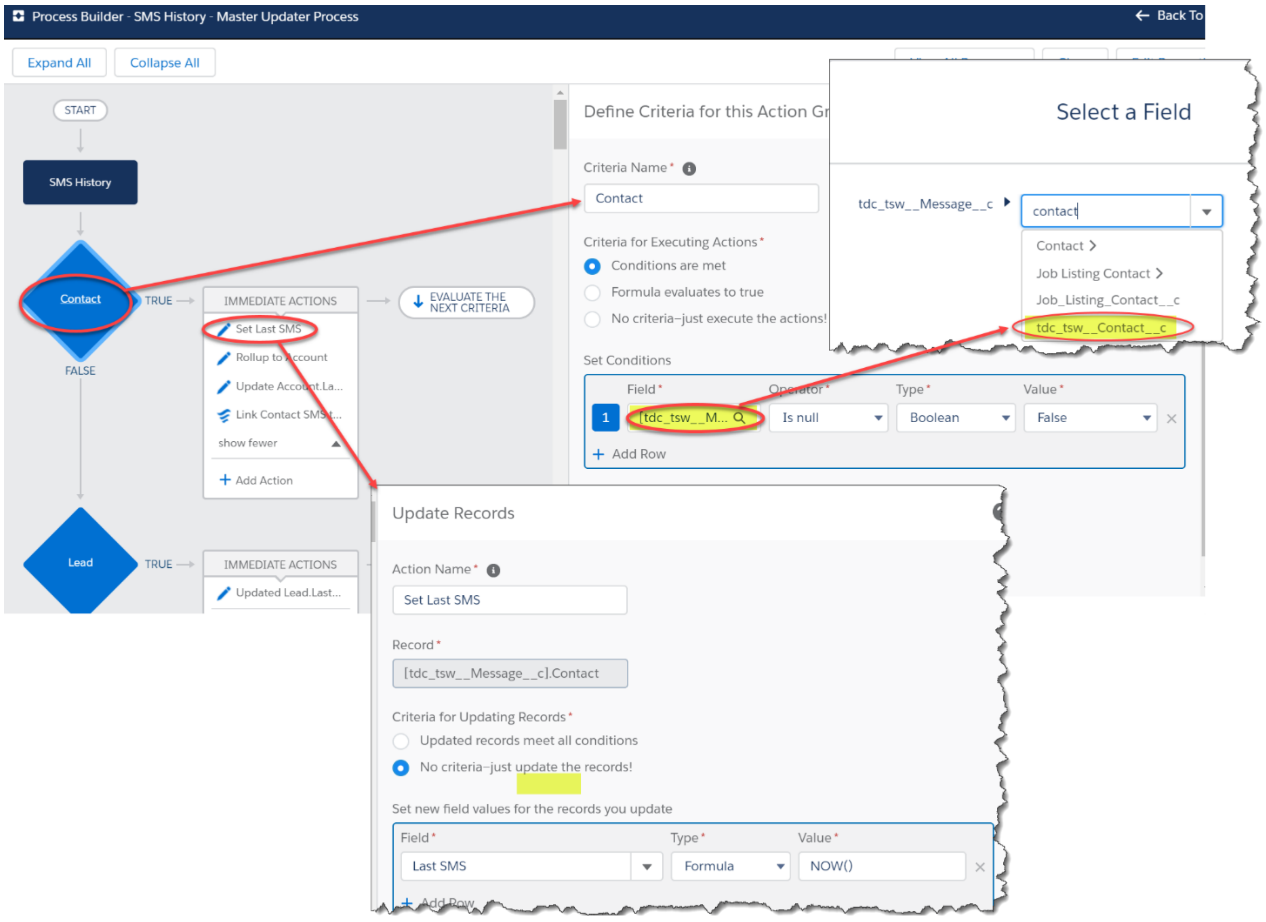
Figure 22 - Classic Roll-Up from a child object to the Contact – set the SMS_History.Contact Id = Job_Listing.Contact_Id

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 differentiate 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 (separate criteria nodes). Then update the respective records Last_SMS field with the SMS_History.Create_Date. Now your field can be used as a filter on List Views.


Process Builder - SMS History - Master Updater Process

Define Criteria for this Action Group

Criteria Name: Contact

Criteria for Executing Actions: Conditions are met

Set Conditions

Field	Operator	Type	Value
[tdc_tsw__M...	Is null	Boolean	False

Select a Field

tdc_tsw__Message__c ▸ contact

Contact >
Job Listing Contact >
Job_Listing_Contact__c
tdc_tsw__Contact__c

Update Records

Action Name: Set Last SMS

Record: [tdc_tsw__Message__c].Contact

Criteria for Updating Records: No criteria–just update the records!

Set new field values for the records you update

Field	Type	Value
Last SMS	Formula	NOW()

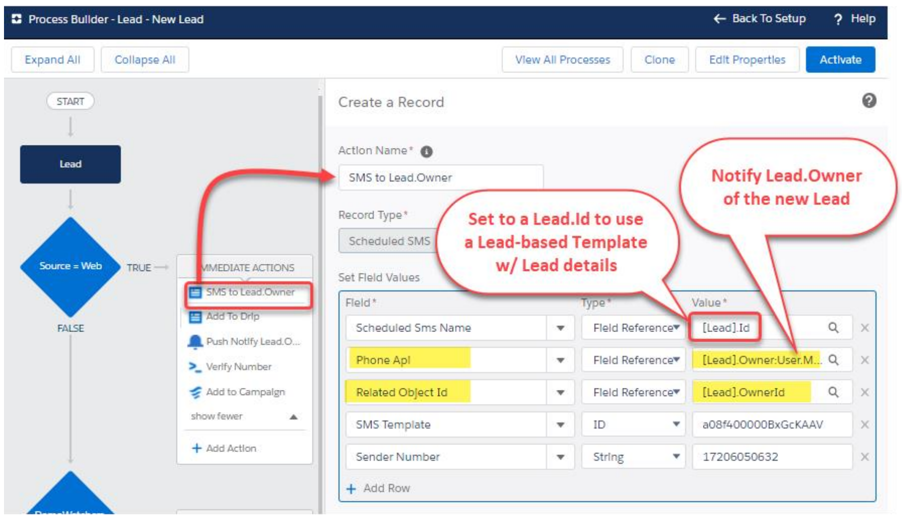
Figure 23 - Set a Contact.Last_SMS field

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.

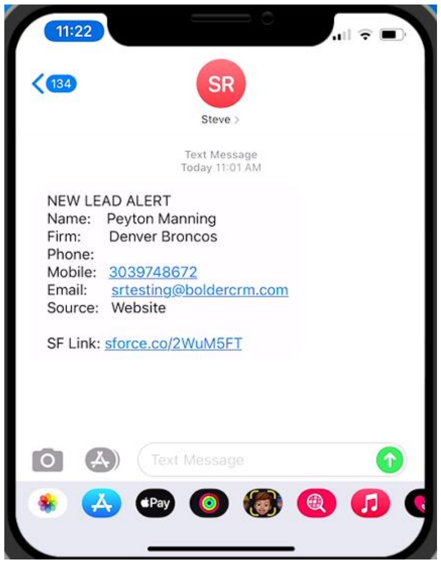


Action Name*: SMS to Lead.Owner

Record Type*: Scheduled SMS

Set Field Values

Field*	Type*	Value*
Scheduled Sms Name	Field Reference	[Lead].Id
Phone Api	Field Reference	[Lead].Owner:User.M...
Related Object Id	Field Reference	[Lead].OwnerId
SMS Template	ID	a08f400000BxGcKAAV
Sender Number	String	17206050632

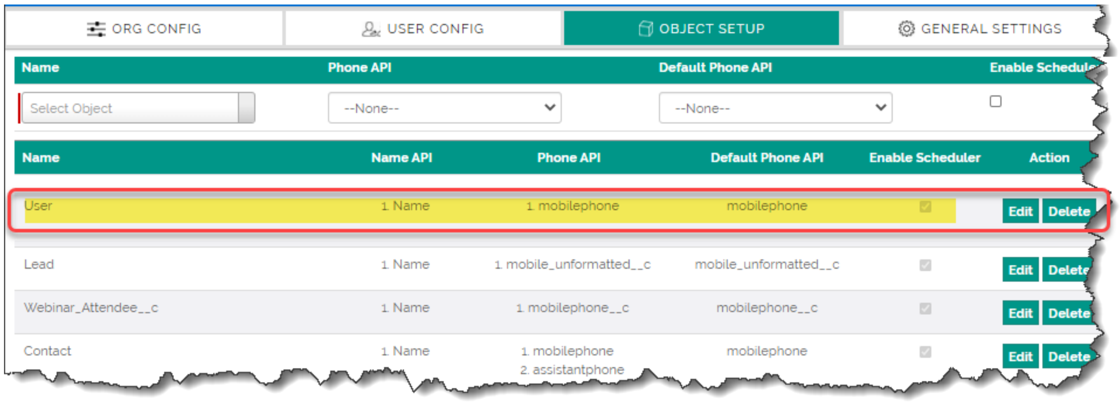


Text Message
Today 11:01 AM

NEW LEAD ALERT
Name: Peyton Manning
Firm: Denver Broncos
Phone:
Mobile: 3039748672
Email: srtesting@boldercrm.com
Source: Website

SF Link: sforce.co/2WuM5FT

Figure 24 - Sending text alerts to internal users dynamically using the Lead.Owner.User.MobilePhone



Name	Phone API	Default Phone API	Enable Schedule
Select Object	--None--	--None--	☐

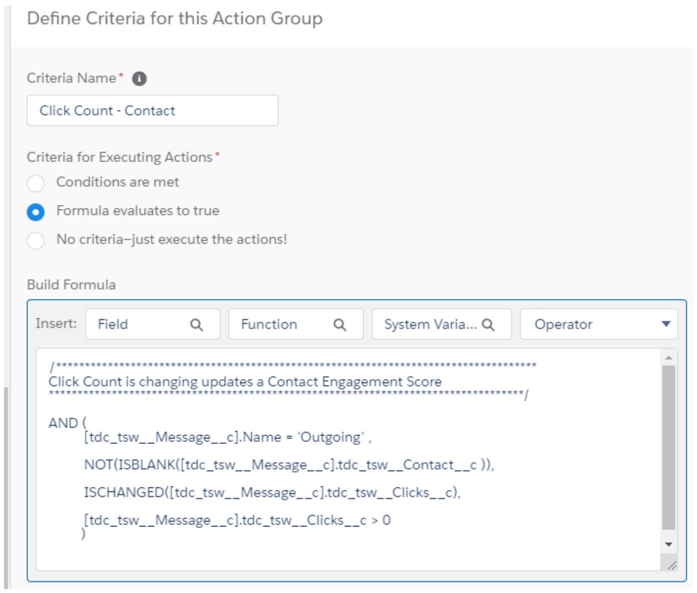
Name	Name API	Phone API	Default Phone API	Enable Scheduler	Action
User	1. Name	1. mobilephone	mobilephone	✓	Edit Delete
Lead	1. Name	1. mobile_unformatted__c	mobile_unformatted__c	✓	Edit Delete
Webinar_Attendee__c	1. Name	1. mobilephone__c	mobilephone__c	✓	Edit Delete
Contact	1. Name	1. mobilephone 2. assistantphone	mobilephone	✓	Edit Delete

Figure 25 - if using this method make sure the USER object is configured for texting

Hyperlink Tracking Actions - Update a Lead Score

The 360SMS platform offers an awesome Hyperlink tracking feature which not only notifies the SMS_History.Owner of the click but can be used in other automations such as incrementing a Lead Score or a Last_Click_Date field, or perhaps creating a Salesforce Call (Task). Below is a simple example of incrementing a Lead Score field but the immediate action could be anything you imagine.

Simply create a process builder on the UPDATE of the SMS History object since the Click Count will be updated automatically each time



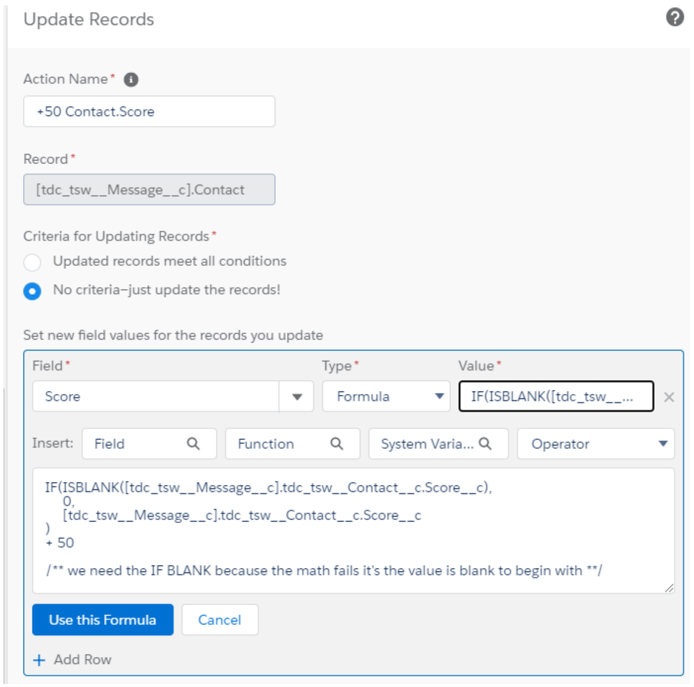
Define Criteria for this Action Group

Criteria Name*: Click Count - Contact

Criteria for Executing Actions*
- Conditions are met
- [✓ Formula evaluates to true]
- No criteria–just execute the actions!

Build Formula

```
/*************************************************************************
Click Count is changing updates a Contact Engagement Score
*************************************************************************/
AND (
     [tdc_tsw__Message__c].Name = 'Outgoing' ,
     NOT(ISBLANK([tdc_tsw__Message__c].tdc_tsw__Contact__c )),
     ISCHANGED([tdc_tsw__Message__c].tdc_tsw__Clicks__c),
     [tdc_tsw__Message__c].tdc_tsw__Clicks__c > 0
     )
```



Update Records

Action Name*: +50 Contact.Score

Record*: [tdc_tsw__Message__c].Contact

Criteria for Updating Records*
- Updated records meet all conditions
- [✓ No criteria–just update the records!]

Set new field values for the records you update

Field*	Type*	Value*
Score	Formula	IF(ISBLANK([tdc_tsw__...

```
IF(ISBLANK([tdc_tsw__Message__c].tdc_tsw__Contact__c.Score__c),
    0,
    [tdc_tsw__Message__c].tdc_tsw__Contact__c.Score__c
)
+ 50
/** we need the IF BLANK because the math fails it's the value is blank to begin with **/
```

Code snippets for easy copy/pasting:

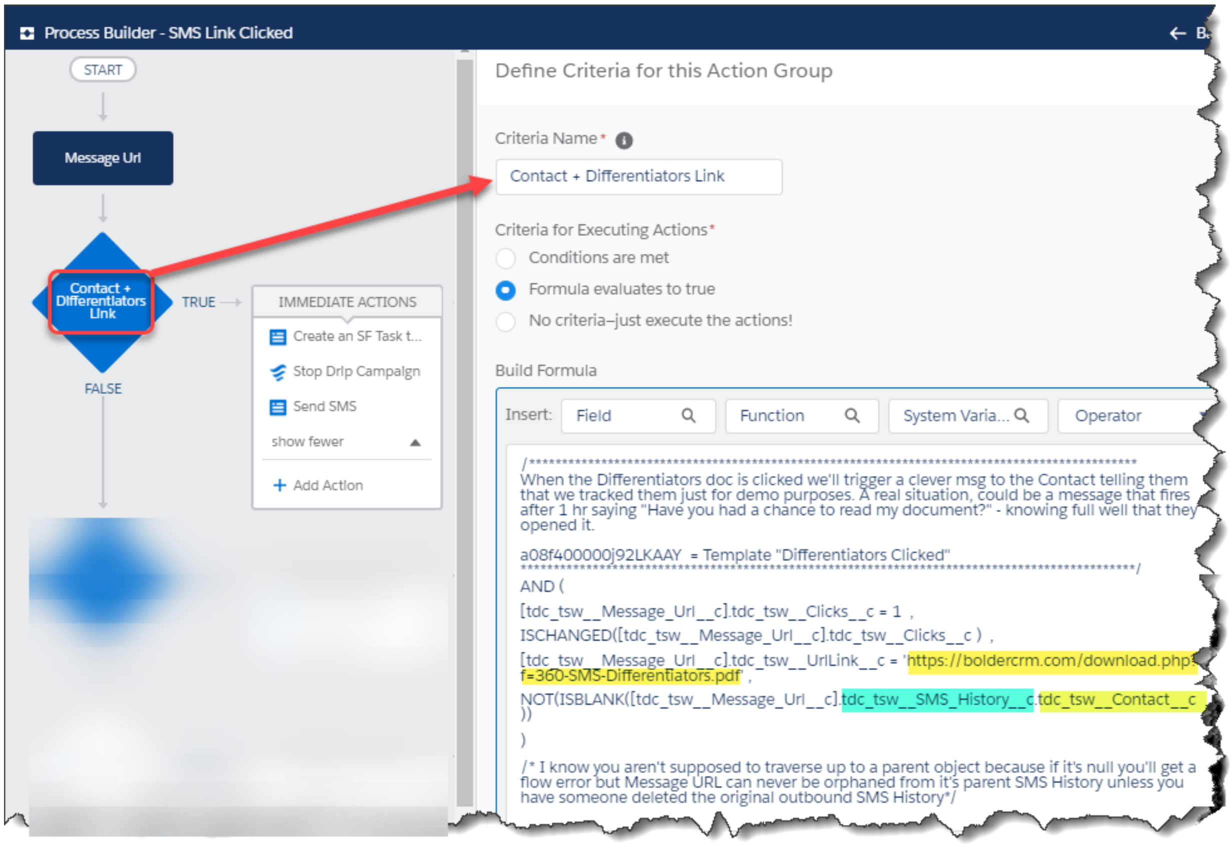
Criteria

```
/*************************************************************************
Click Count is changing updates a Contact Engagement Score
*************************************************************************/
AND (
     [tdc_tsw__Message__c].Name = 'Outgoing' ,
     NOT(ISBLANK([tdc_tsw__Message__c].tdc_tsw__Contact__c )),
     ISCHANGED([tdc_tsw__Message__c].tdc_tsw__Clicks__c),
     [tdc_tsw__Message__c].tdc_tsw__Clicks__c > 0
     )
```

Immediate Action

```
/** We need the BLANK check because the math doesn't work when it's blank to add 50 **/
IF(  ISBLANK([tdc_tsw__Message__c].tdc_tsw__Contact__c.Score__c),
     0,
     [tdc_tsw__Message__c].tdc_tsw__Contact__c.Score__c
)
+ 50
```

If you are wanting to take action on a specific URL, currently the SMS History object only has the SMS_History.**URL** field which unfortunately is the Bitly or TinyURL that is unique to each customer. Therefore, you must trigger on the **Message URL** object (child of SMS History holding the many URLS and actual fields for tracking). We've requested that the **UrlLink** field be moved up to the SMS History.



Define Criteria for this Action Group

Criteria Name* Contact + Differentiators Link

Criteria for Executing Actions*

Conditions are met

[✓ Formula evaluates to true]

No criteria—just execute the actions!

Build Formula

Insert: Field | Function | System Varia... | Operator

```
/************************************************************************************
When the Differentiators doc is clicked we'll trigger a clever msg to the Contact telling them
that we tracked them just for demo purposes. A real situation, could be a message that fires
after 1 hr saying "Have you had a chance to read my document?" - knowing full well that they
opened it.

a08f400000j92LKAAY  = Template "Differentiators Clicked"
************************************************************************************/
AND (
[tdc_tsw__Message_Url__c].tdc_tsw__Clicks__c = 1 ,
ISCHANGED([tdc_tsw__Message_Url__c].tdc_tsw__Clicks__c ) ,
[tdc_tsw__Message_Url__c].tdc_tsw__UrlLink__c = 'https://boldercrm.com/download.php?
f=360-SMS-Differentiators.pdf' ,
NOT(ISBLANK([tdc_tsw__Message_Url__c].tdc_tsw__SMS_History__c.tdc_tsw__Contact__c
))
)
/* I know you aren't supposed to traverse up to a parent object because if it's null you'll get a
flow error but Message URL can never be orphaned from it's parent SMS History unless you
have someone deleted the original outbound SMS History*/
```

Criteria above for easy copy/pasting

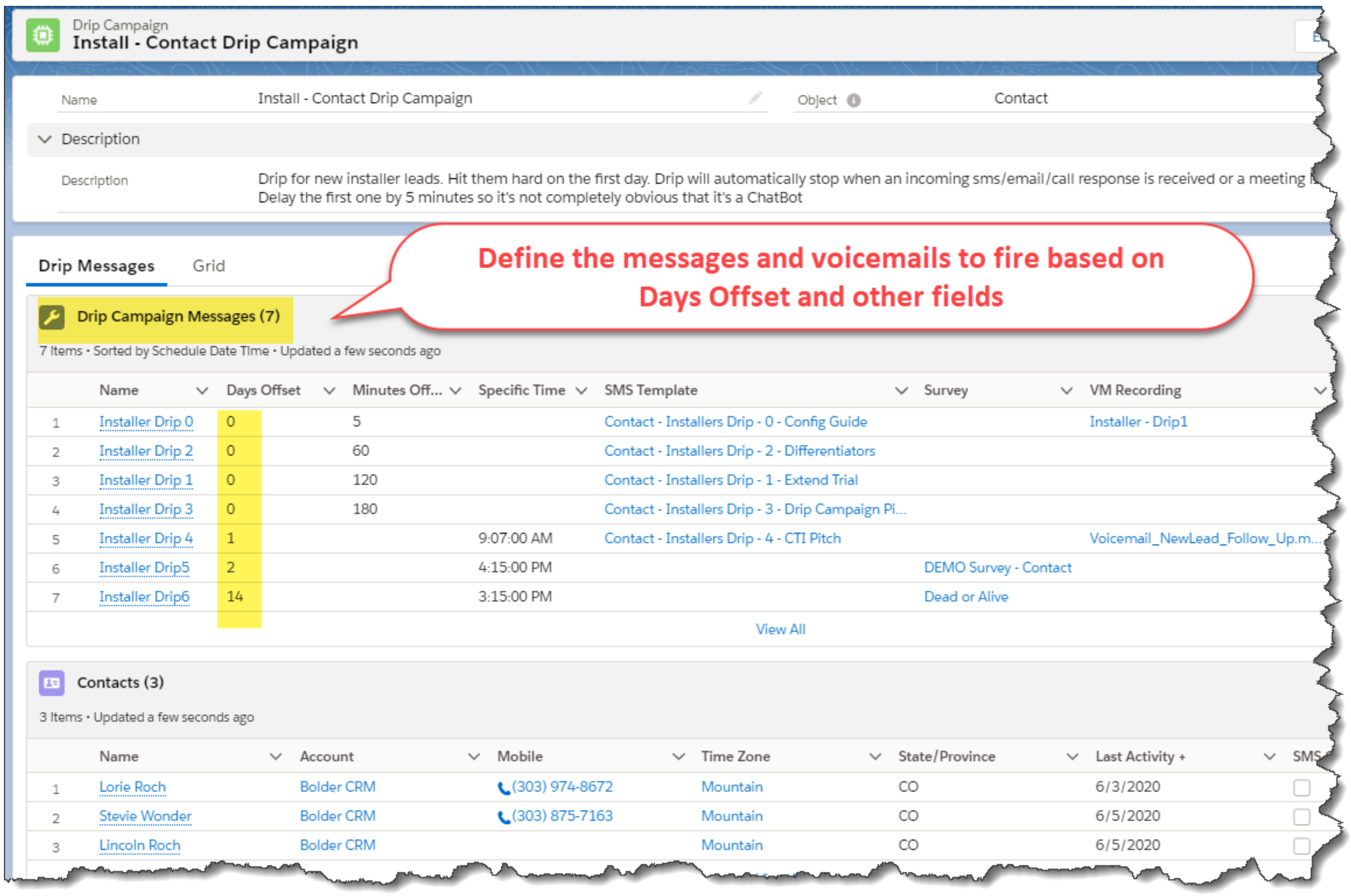
```
/*****************************************************************************************
When the Differentiators doc is clicked, we'll trigger a clever msg to the Contact telling them
that we tracked them just for demo purposes. A real situation could be a message that fires
after 1 hr saying "Have you had a chance to read my document?" - knowing full well that they
opened it.

SMS Template to Fire:  a08f400000j92LKAAY = "Differentiators Clicked"
*****************************************************************************************/

AND (
     [tdc_tsw__Message_Url__c].tdc_tsw__Clicks__c = 1  ,
     ISCHANGED([tdc_tsw__Message_Url__c].tdc_tsw__Clicks__c )  ,
     NOT(ISBLANK([tdc_tsw__Message_Url__c].tdc_tsw__SMS_History__c.tdc_tsw__Contact__c )) ,

     [tdc_tsw__Message_Url__c].tdc_tsw__UrlLink__c = 'https://boldercrm.com/download.php?f=360-
SMS-Differentiators.pdf'
)

/* I know you aren't supposed to traverse up to a parent object because if it's null you'll get
a flow error but Message URL can never be orphaned from its parent SMS History unless you have
somehow deleted the original outbound SMS History*/
```

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 module (purchased separately). 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. Read the **SMS Drip Campaigns** documentation for the programmatic methods of starting and stopping Drip Campaigns.



Drip Campaign
Install - Contact Drip Campaign

Name: Install - Contact Drip Campaign
Object: Contact

Description

Description: Drip for new installer leads. Hit them hard on the first day. Drip will automatically stop when an incoming sms/email/call response is received or a meeting
Delay the first one by 5 minutes so it's not completely obvious that it's a ChatBot

Drip Messages | Grid

Define the messages and voicemails to fire based on Days Offset and other fields

Drip Campaign Messages (7)
7 Items • Sorted by Schedule Date Time • Updated a few seconds ago

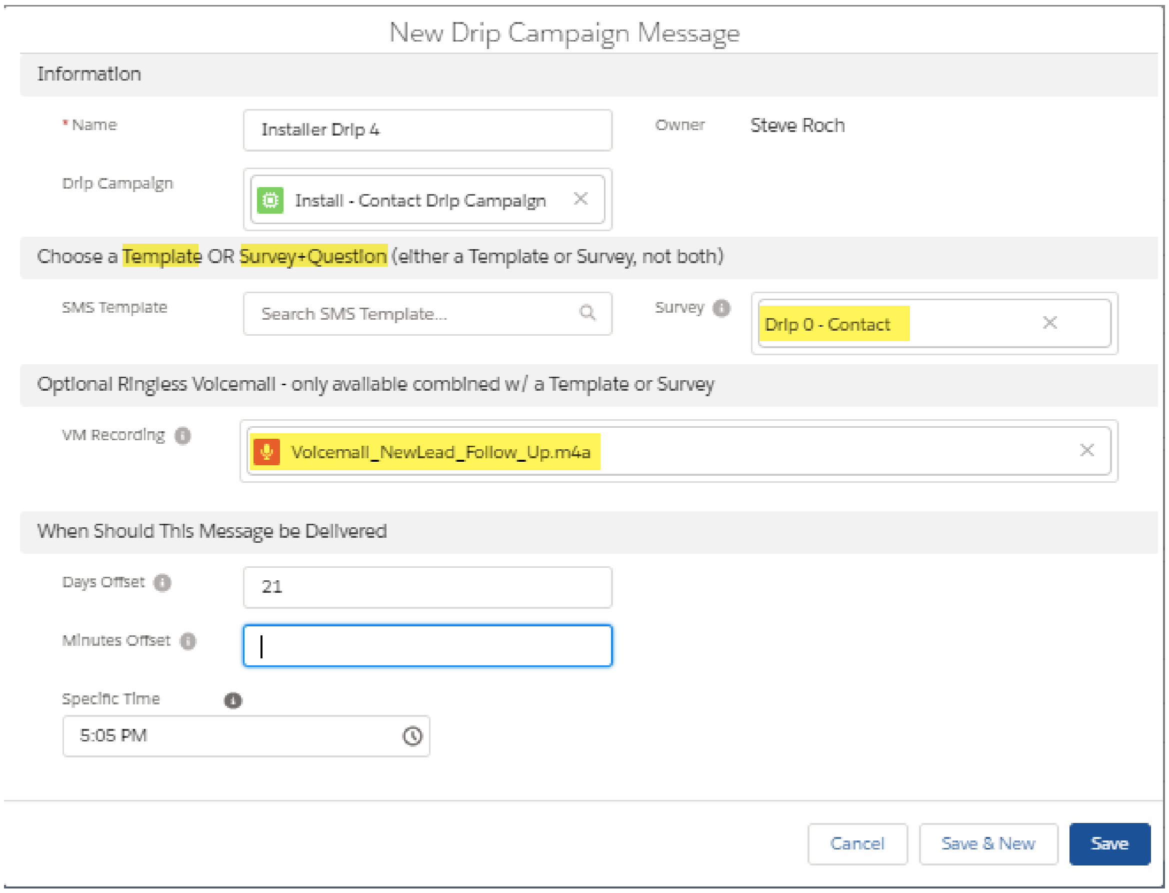
	Name	Days Offset	Minutes Off...	Specific Time	SMS Template	Survey	VM Recording
1	Installer Drip 0	0	5		Contact - Installers Drip - 0 - Config Guide		Installer - Drip1
2	Installer Drip 2	0	60		Contact - Installers Drip - 2 - Differentiators		
3	Installer Drip 1	0	120		Contact - Installers Drip - 1 - Extend Trial		
4	Installer Drip 3	0	180		Contact - Installers Drip - 3 - Drip Campaign Pi...		
5	Installer Drip 4	1		9:07:00 AM	Contact - Installers Drip - 4 - CTI Pitch		Voicemail_NewLead_Follow_Up.m...
6	Installer Drip5	2		4:15:00 PM		DEMO Survey - Contact	
7	Installer Drip6	14		3:15:00 PM		Dead or Alive	

View All

Contacts (3)
3 Items • Updated a few seconds ago

	Name	Account	Mobile	Time Zone	State/Province	Last Activity	SMS
1	Lorie Roch	Bolder CRM	(303) 974-8672	Mountain	CO	6/3/2020	☐
2	Stevie Wonder	Bolder CRM	(303) 875-7163	Mountain	CO	6/5/2020	☐
3	Lincoln Roch	Bolder CRM		Mountain	CO	6/5/2020	☐

Figure 26 - Drip Campaign are defined by a series of SMS Templates or Surveys to fire at different Days and Times.



New Drip Campaign Message

Information

* Name	Installer Drip 4	Owner	Steve Roch
Drip Campaign	Install - Contact Drip Campaign		

Choose a Template OR Survey+Question (either a Template or Survey, not both)

SMS Template	Search SMS Template...	Survey	Drip 0 - Contact

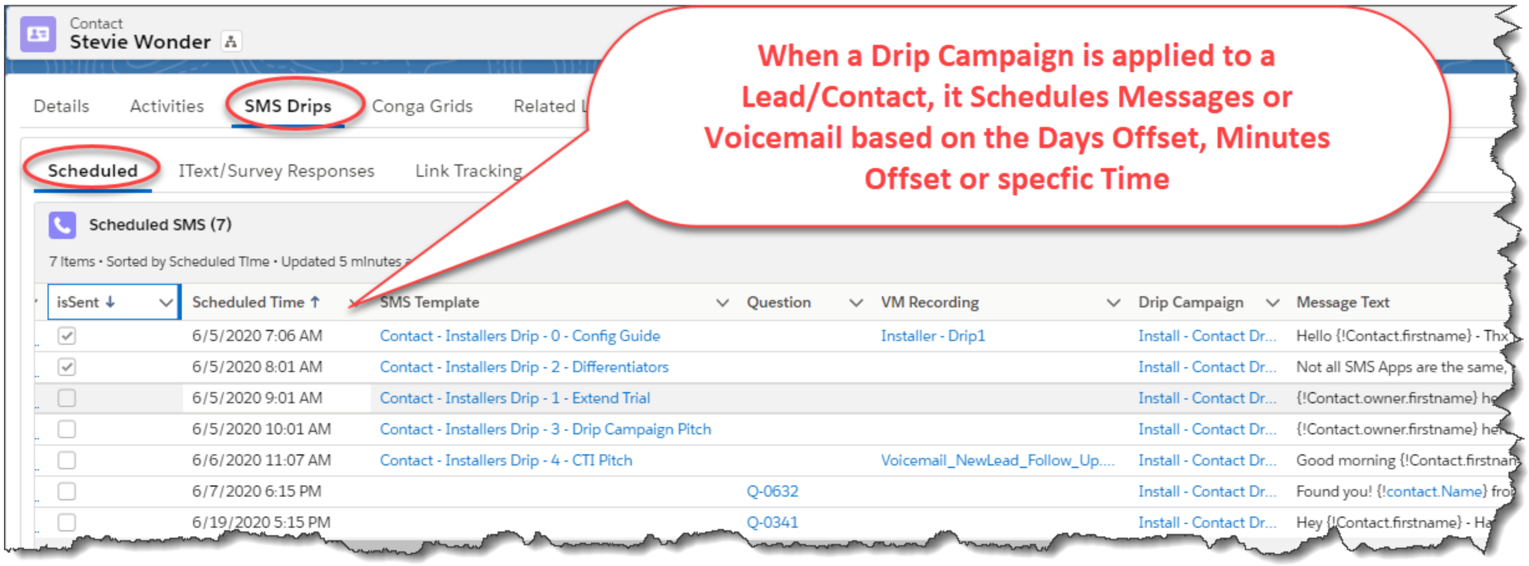
Optional Ringless Voicemail - only available combined w/ a Template or Survey

VM Recording	Voicemail_NewLead_Follow_Up.m4a

When Should This Message be Delivered

Days Offset	21
Minutes Offset	
Specific Time	5:05 PM

*Figure 27 – Define each days Template or Survey by defining the **days offset** from the day it is applied + an optional time of day*



Contact
Stevie Wonder

When a Drip Campaign is applied to a Lead/Contact, it Schedules Messages or Voicemail based on the Days Offset, Minutes Offset or specfic Time

Scheduled SMS (7)

7 Items • Sorted by Scheduled Time • Updated 5 minutes

isSent	Scheduled Time	SMS Template	Question	VM Recording	Drip Campaign	Message Text
✓	6/5/2020 7:06 AM	Contact - Installers Drip - 0 - Config Guide		Installer - Drip1	Install - Contact Dr...	Hello {!Contact.firstname} - Thx
✓	6/5/2020 8:01 AM	Contact - Installers Drip - 2 - Differentiators			Install - Contact Dr...	Not all SMS Apps are the same,
☐	6/5/2020 9:01 AM	Contact - Installers Drip - 1 - Extend Trial			Install - Contact Dr...	{!Contact.owner.firstname} he
☐	6/5/2020 10:01 AM	Contact - Installers Drip - 3 - Drip Campaign Pitch			Install - Contact Dr...	{!Contact.owner.firstname} he
☐	6/6/2020 11:07 AM	Contact - Installers Drip - 4 - CTI Pitch		Voicemail_NewLead_Follow_Up....	Install - Contact Dr...	Good morning {!Contact.firstnan
☐	6/7/2020 6:15 PM		Q-0632		Install - Contact Dr...	Found you! {!contact.Name} fro
☐	6/19/2020 5:15 PM		Q-0341		Install - Contact Dr...	Hey {!Contact.firstname} - Ha

Figure 28 - Drip Campaign Applied creates Scheduled SMS records at the given Days Offset + Time of Day (if specified)

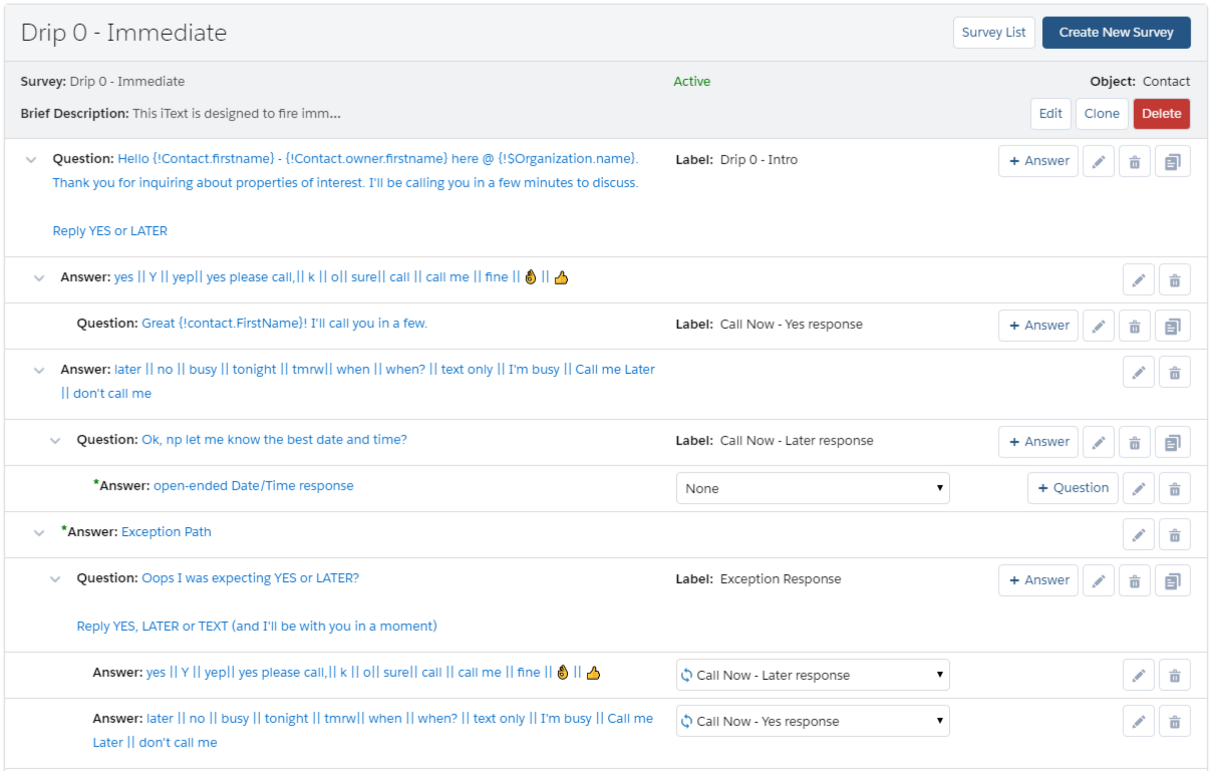
iText, Surveys, ChatBots (iText – Intelligent Texting)

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 but when using an iText/Survey that dialog is additionally stored in a **Survey Response** object for even greater reporting and automation potential. Survey's allow a question to have multiple answer paths which then branch to additional questions with their own multiple answer paths 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." However, it works well for open-ended answers as well, e.g. "Reply with your desired Loan Amount (numeric answer only please)"

Lastly, survey replies can easily update field values in Salesforce objects or take other actions without Process Builders or Flows using the **Survey Response Action Handler** module from **Bolder CRM** or do it yourself with Process Builders/Flows on the SMS History or Survey Response object by inspecting the incoming answer to a specific question. Read more @ Surveys and Incoming Keyword Processing



Survey: Drip 0 - Immediate Active **Object:** Contact

Brief Description: This iText is designed to fire imm...

Question: Hello {!Contact.firstname} - {!Contact.owner.firstname} here @ {!SOrganization.name}. Thank you for inquiring about properties of interest. I'll be calling you in a few minutes to discuss.

Reply YES or LATER — **Label:** Drip 0 - Intro

Answer: yes || Y || yep|| yes please call,|| k || o|| sure|| call || call me || fine || 👌 || 👍

Question: Great {!contact.FirstName}! I'll call you in a few. — **Label:** Call Now - Yes response

Answer: later || no || busy || tonight || tmrw|| when || when? || text only || I'm busy || Call me Later || don't call me

Question: Ok, np let me know the best date and time? — **Label:** Call Now - Later response

***Answer:** open-ended Date/Time response — None

***Answer:** Exception Path

Question: Oops I was expecting YES or LATER?

Reply YES, LATER or TEXT (and I'll be with you in a moment) — **Label:** Exception Response

Answer: yes || Y || yep|| yes please call,|| k || o|| sure|| call || call me || fine || 👌 || 👍 — Call Now - Later response

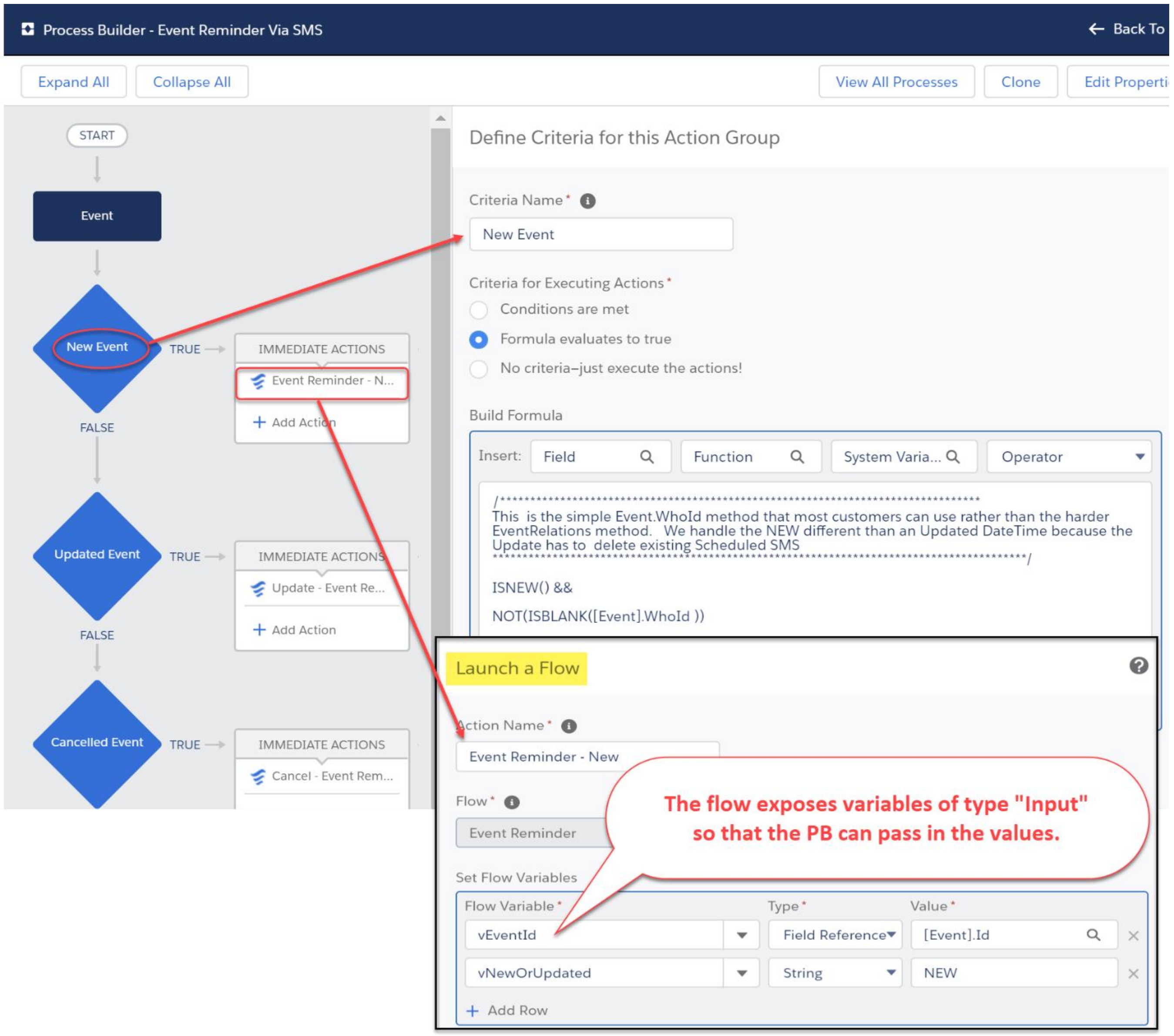
Answer: later || no || busy || tonight || tmrw|| when || when? || text only || I'm busy || Call me Later || don't call me — Call Now - Yes response

Figure 29 - Typical Chatbot Survey sent to a new lead

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, more robust formulas, lookup records and recordsets, loop through records, delete records and more.

Below we demonstrate a more powerful and easier method for **Event Reminders** documented in the SMS Event Reminders guide. With flows we can easily workaround the limitations of the Salesforce **EVENT** object and best of all we can COPY/PASTE components and document our code!



Criteria Name: New Event

Criteria for Executing Actions:
- Conditions are met
- [✓ Formula evaluates to true]
- No criteria–just execute the actions!

Build Formula

```
/*****************************************************************************
This  is the simple Event.WhoId method that most customers can use rather than the harder
EventRelations method.   We handle the NEW different than an Updated DateTime because the
Update has to  delete existing Scheduled SMS
*****************************************************************************/

ISNEW() &&
NOT(ISBLANK([Event].WhoId ))
```

Action Name: Event Reminder - New

Flow: Event Reminder

Set Flow Variables

Flow Variable	Type	Value
vEventId	Field Reference	[Event].Id
vNewOrUpdated	String	NEW

Add Row

Figure 30 - Example of an SMS_History Incoming process builder triggering a call to a flow.

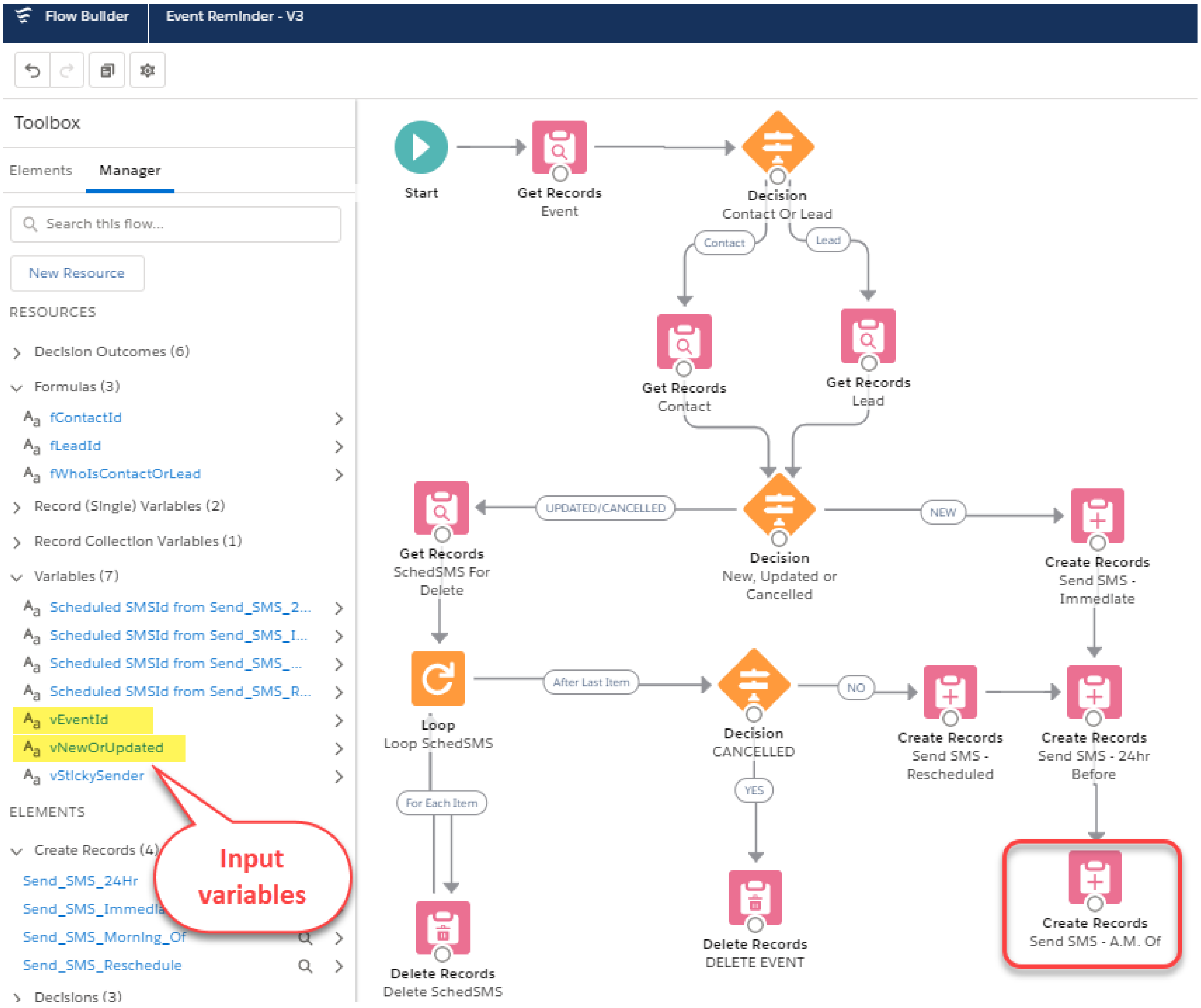
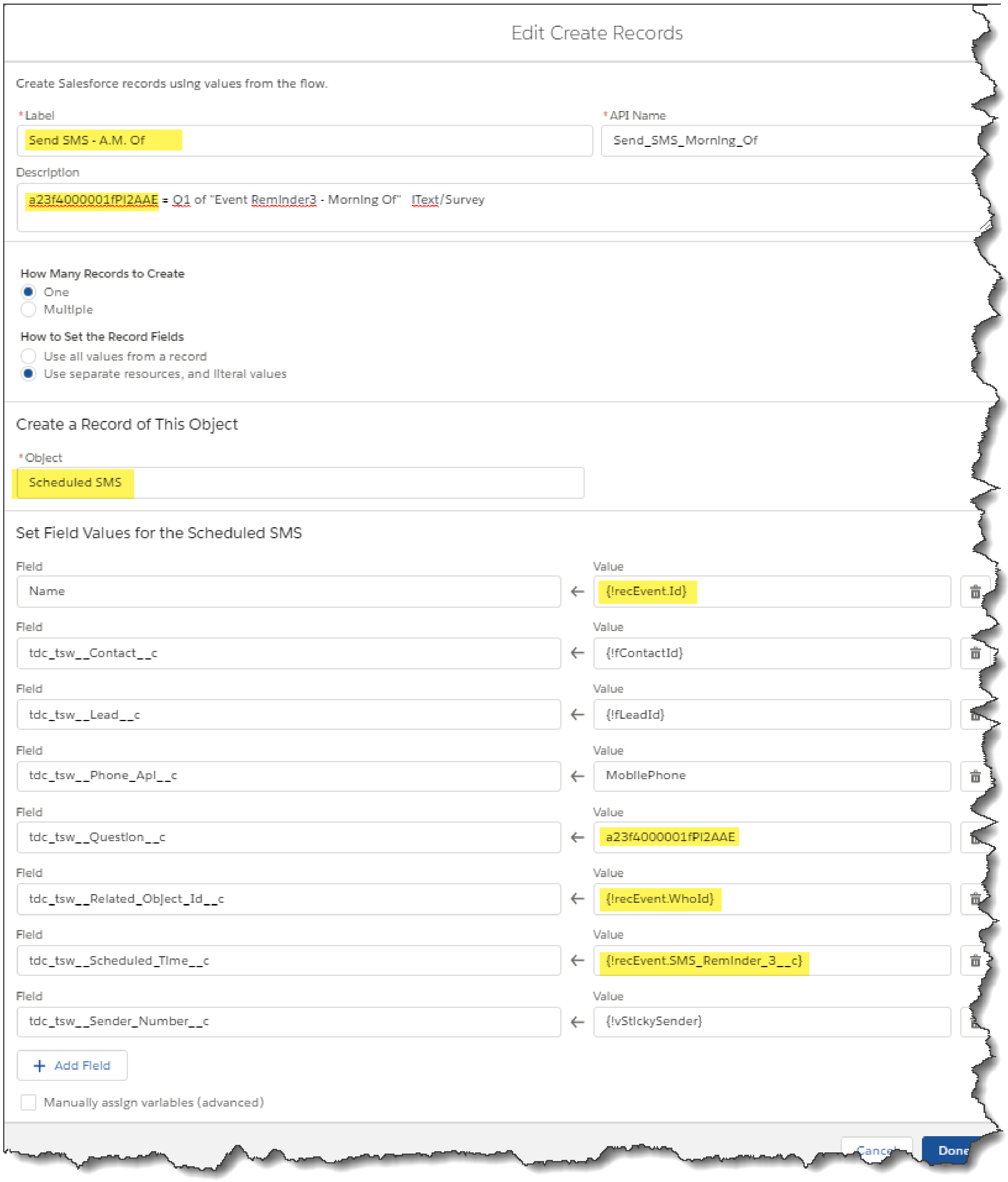


Figure 31 – A classic flow w/ Lookups (Get Records), complex Decision logic, Loops, Delete Records and Create Records for the Scheduled SMS's

With the more powerful capabilities of FLOWS we can handle logic for a New, Updated or Cancelled EVENT. We can easily handle the problems with the Salesforce Event.WhoId's polymorphic (where Process Builder will not let us traverse up to the Contact/Lead). And lastly, we can loop through the Scheduled SMS records to delete them if needed. **Contact Steve directly if you want this Flow for your own Org.**



Edit Create Records

Create Salesforce records using values from the flow.

*Label	*API Name
Send SMS - A.M. Of	Send_SMS_Morning_Of

Description

a23f4000001fPI2AAE = Q1 of "Event Reminder3 - Morning Of"   IText/Survey

How Many Records to Create

- (●) One
- ( ) Multiple

How to Set the Record Fields

- ( ) Use all values from a record
- (●) Use separate resources, and literal values

Create a Record of This Object

*Object

Scheduled SMS

Set Field Values for the Scheduled SMS

Field		Value
Name	←	{!recEvent.Id}
tdc_tsw__Contact__c	←	{!fContactId}
tdc_tsw__Lead__c	←	{!fLeadId}
tdc_tsw__Phone_Api__c	←	MobilePhone
tdc_tsw__Question__c	←	a23f4000001fPI2AAE
tdc_tsw__Related_Object_Id__c	←	{!recEvent.WhoId}
tdc_tsw__Scheduled_Time__c	←	{!recEvent.SMS_Reminder_3__c}
tdc_tsw__Sender_Number__c	←	{!vStickySender}

+ Add Field

☐ Manually assign variables (advanced)

Cancel Done

Figure 32 - The creation of the Scheduled SMS is the same as a Process Builder – but you can COPY/PASTE components in FLOWS!!!!!   And you can use the DESCRIPTION in every component to document your code!

Troubleshooting SMS Process Builders, Flows and Scheduled SMS

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 or flows. Consulting agreements are available though.

1. Always assume that your SMS is not firing because of a **problem with your business logic**, its parameters or with the user or sender number 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 incorrect business logic for 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 eliminate the criteria as the culprit first.

3. **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 this user in the User Config table. This is the #1 source of "my SMS is not firing"

4. **Malformed or Invalid Sender Number** – Commonly we'll see Scheduled SMS's look as if they are sent (isSent=True) but no SMS_History. The most common culprit is an invalid or malformed SenderNumber. Note that the value of **Sticky_Sender** for most orgs is a formula based on the field Record.SMS_Last_Sender_Number__c or if this is value is null it will get the number from Record.Owner → User.Phone. Thus, the most common problem is a User.Phone that is not a valid SenderNumber (often it will hold their personal cell phone number) or someone has imported Leads with a Last Sender Number that is not a proper 360 Sender Number in the proper format of 19998887777. The number must have the country code (1 for USA) and have no punctuation. **Note:** The triggering user still must have access to the SenderNumber (SMS User Config) either explicitly or via the All Users (see item 3 above).

5. **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)` (i.e. Salesforce date math is always in days so: 1 min ÷ 60 min/hr ÷ 24hrs/day)

6. **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.

7. **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 or (Survey) Question parameter. Object Id + Template Id mismatches are common problems.

8. **Integration Users** – commonly customers will have integration processes like Pardot/Marketo 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 people commonly forget to create the necessary User Config record associating the User + Sender Number.

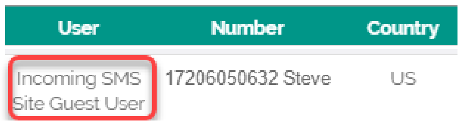
   a. You may need to Contact technical support as licensing for Site Guest Users and other integration users can sometimes be complicated due to how Salesforce treats these "users."

9. **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. This is especially the case when triggering off of the SMS History or Survey Response objects where the App User is always the CreatedBy or LastModified User.

   b. 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.

   c. **Hyperlink Click Triggers** – If triggering off the SMS History or **Message URL** object (the object that holds the click count to trackable links sent to customers), pay special that this user is the special "**Incoming SMS Site Guest User**" created when Hyperlink tracking was set up for your org. In other words, when a hyperlink is clicked this is the user that updates the click count. That user is already licensed but admins commonly forget to associate this user to a one or more sender numbers.



User	Number	Country
Incoming SMS Site Guest User	17206050632 Steve	US

10. **Salesforce Record Ids** - When using Salesforce record ID values in your criteria be aware that Salesforce Classic will often present only the 15 character ID of the record in the URL whereas the actual value is almost always stored in the database as an 18 character ID.

a. This is common when detecting an Inbound SMS responding to a particular Survey Question or Template where you might be taking some sort of action on the answer based on that particular question/template (since the Incoming SMS inherits it's previous Outbound QuestionId or TemplateId).

b. Either always compare an 18-character Id to an 18-Character Id or compare the first 15 characters to a 15-character Id.
`LEFT(somefield, 15) = '003f400000P3noE` - where we've stripped off the last 3 characters

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 and now renamed **Conga Grid™**. 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/Text:	+1 720.605.0632 or +44 1234 480 564 or +61 488 845 944
WhatsApp:	+1 303.800.3258 or click here initiate a WhatsApp convo with Steve
Email:	steve@boldercrm.com