I

360 SMS APP

Sending SMS through Flows

360 DEGREE CLOUD TECHNOLOGIES PVT. LTD.

Contents

Sending SMS Through Flows Basics 3

Another way to create record trigger flow 8

Relating Outbound SMS to an Alternate Object 13

Method #2 – Apex Class 14

Method #3 – Apex Triggers 17

Send MMS Via Flows: 19

Dynamic Sender Number 22

Record based Sender Number (Sticky Sender) 23

Dynamic Sender Number – Record Owner 25

Dynamic Sender Number – Geography or Business Rule Based 26

Dynamic MMS – such as sending a picture of a particular user 27

Dark Hours 28

Roll-up SMS Conversations to Parent Objects 29

Create a Contact/Lead Last_SMS field 31

Triggered Texting to Internal Users 32

Create a Master Send SMS Handler 33

iTexts 35

Using Salesforce Flows 36

Sending SMS Through Flows Basics

Salesforce Flows provide a code-free approach for effectively managing the triggering of Outbound Text Messages and handling Incoming Messages based on Keywords or other criteria. It offers the flexibility to trigger actions on any Object within the Salesforce platform.

Lead/Contact: These are commonly used scenarios where Outbound SMS messages are triggered when leads/Contact are created or when specific fields undergo changes.

Custom Objects: Similar to Lead/Contact cases, 360 SMS supports triggered messages from any custom Object. Its SMS Templates and iTexts are compatible with all custom Objects.

SMS_History: This is particularly valuable for handling incoming SMS messages. It allows you to read the message and take subsequent actions based on the content, such as updating the corresponding Salesforce record or sending out additional questions. This functionality is useful for conducting iTexts, where a reply of "INTERESTED" or "NO" can trigger updates to fields or statuses in the associated Salesforce record.

To trigger a message, you only need to configure a few straightforward settings:

For immediate action, set "Create a Record" to "Scheduled SMS" and then specify the following parameters either through Flow:

Please note that only the parameters listed below are allowed. It's important to emphasize that setting the "OWNER" or "Message_Text" fields is not possible, despite them being exposed in the Scheduled SMS Object. However, it allows for providing a custom string while requiring the use of either an SMS_Template_Id or a Question_Id.

To Create the Flow follow the below steps:

Go to Setup>> Search for “Flows” on Quick Action>> Click on “New Flow” >> Choose the Flow Type>> And here you go, you can create the flow now as per the requirement

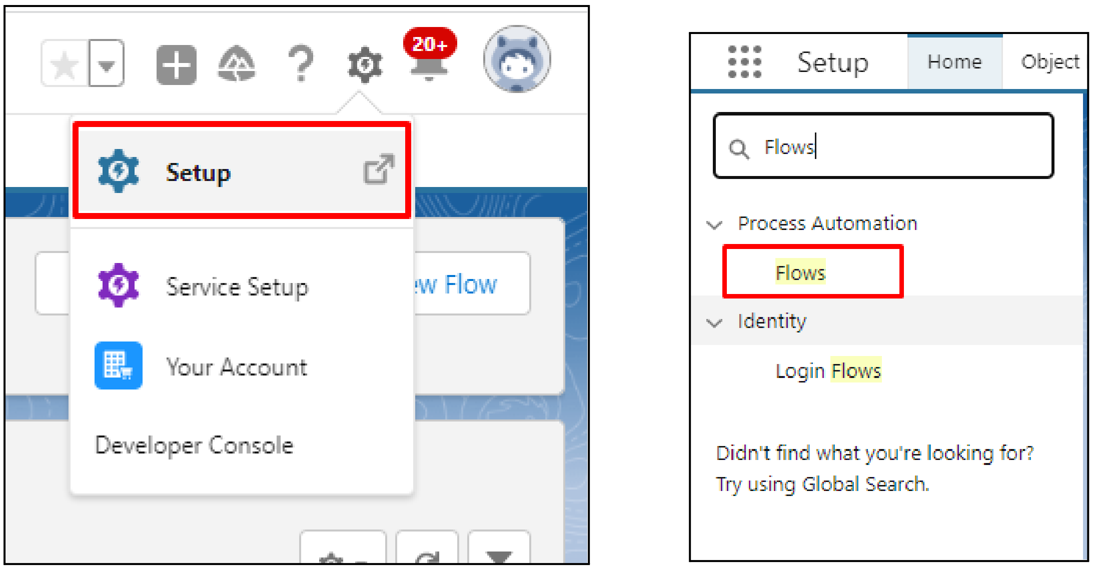


Fig: 1 Setup Option and Flow in Quick search

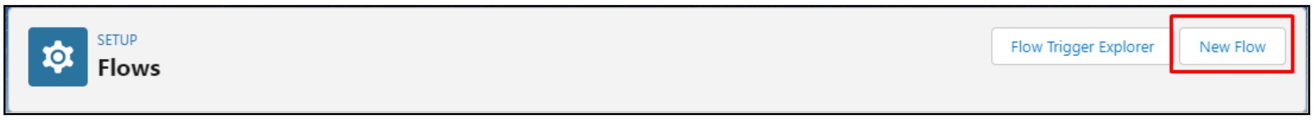
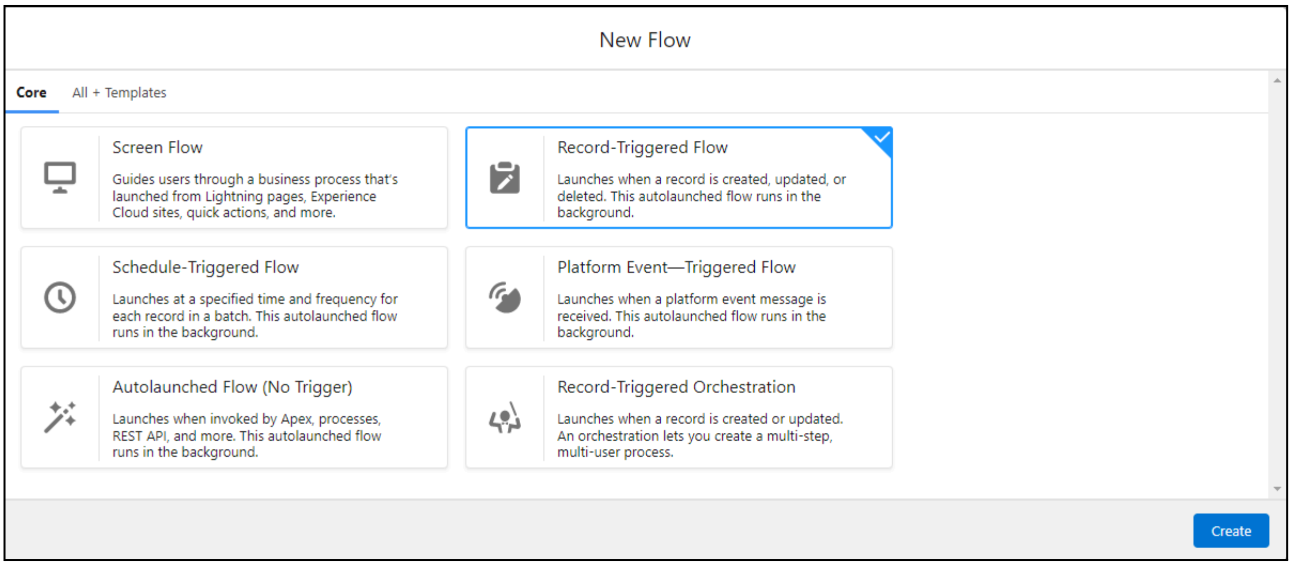


Fig: 2 New Flow Button



Screen Flow
Guides users through a business process that's launched from Lightning pages, Experience Cloud sites, quick actions, and more.

Record-Triggered Flow
Launches when a record is created, updated, or deleted. This autolaunched flow runs in the background.

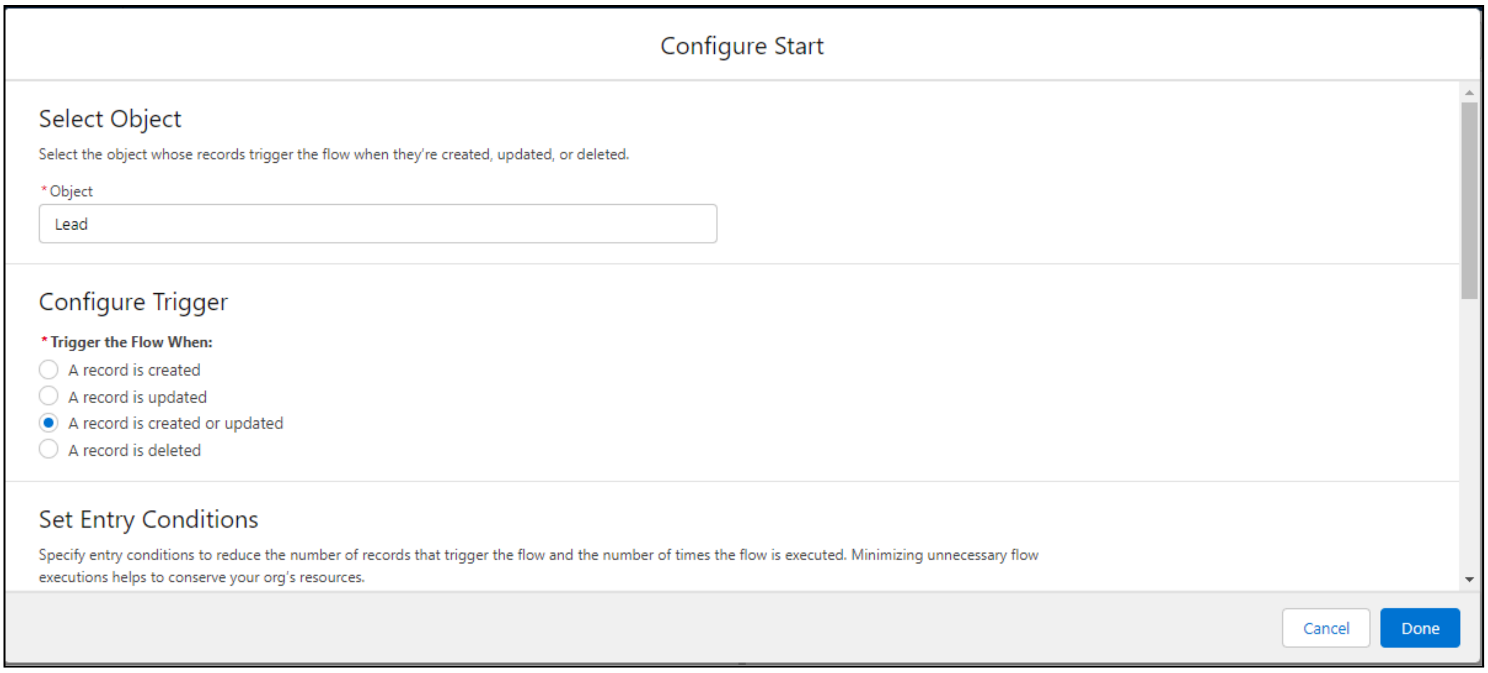
Schedule-Triggered Flow
Launches at a specified time and frequency for each record in a batch. This autolaunched flow runs in the background.

Platform Event—Triggered Flow
Launches when a platform event message is received. This autolaunched flow runs in the background.

Autolaunched Flow (No Trigger)
Launches when invoked by Apex, processes, REST API, and more. This autolaunched flow runs in the background.

Record-Triggered Orchestration
Launches when a record is created or updated. An orchestration lets you create a multi-step, multi-user process.

Fig: 3 Flow type



Configure Start

Select Object

Select the object whose records trigger the flow when they're created, updated, or deleted.

* Object: Lead

Configure Trigger

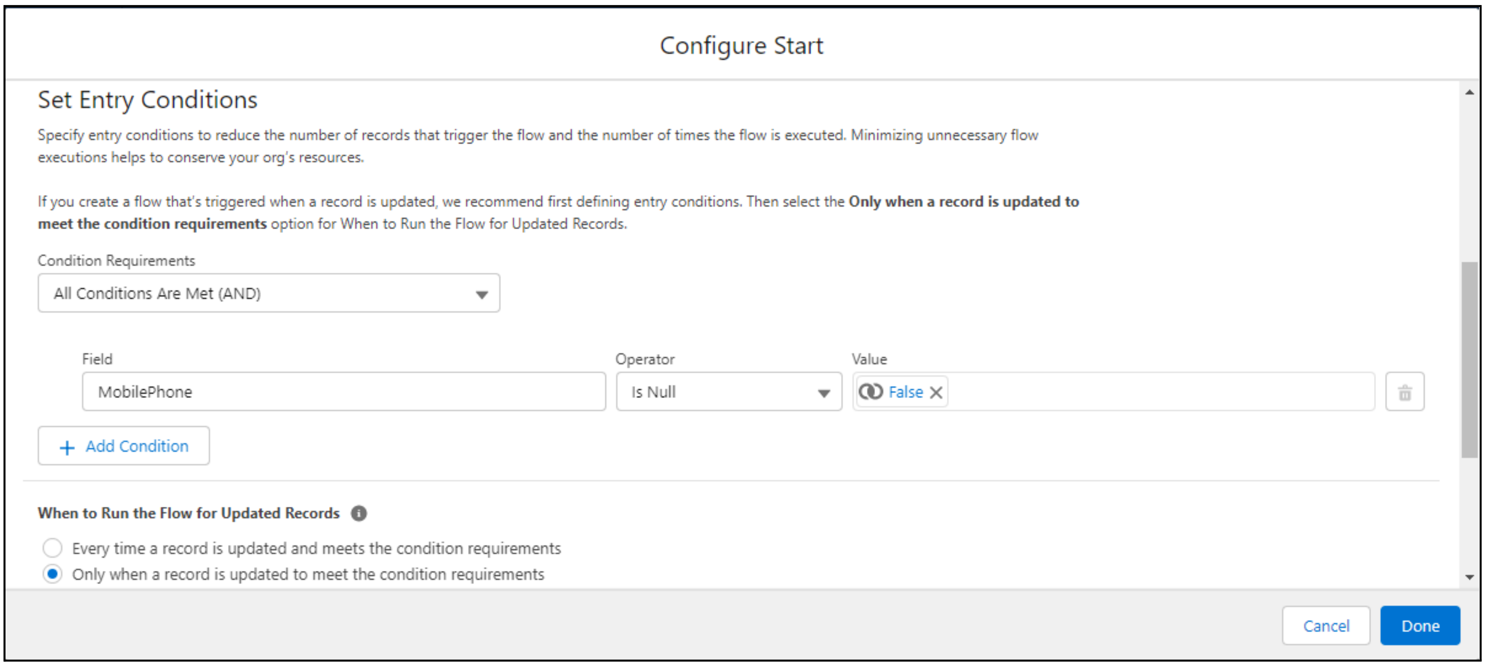
* **Trigger the Flow When:**

- ○ A record is created
- ○ A record is updated
- ● A record is created or updated
- ○ A record is deleted

Set Entry Conditions

Specify entry conditions to reduce the number of records that trigger the flow and the number of times the flow is executed. Minimizing unnecessary flow executions helps to conserve your org's resources.

Fig: 4 Configure a flow



Configure Start

Set Entry Conditions

Specify entry conditions to reduce the number of records that trigger the flow and the number of times the flow is executed. Minimizing unnecessary flow executions helps to conserve your org's resources.

If you create a flow that's triggered when a record is updated, we recommend first defining entry conditions. Then select the **Only when a record is updated to meet the condition requirements** option for When to Run the Flow for Updated Records.

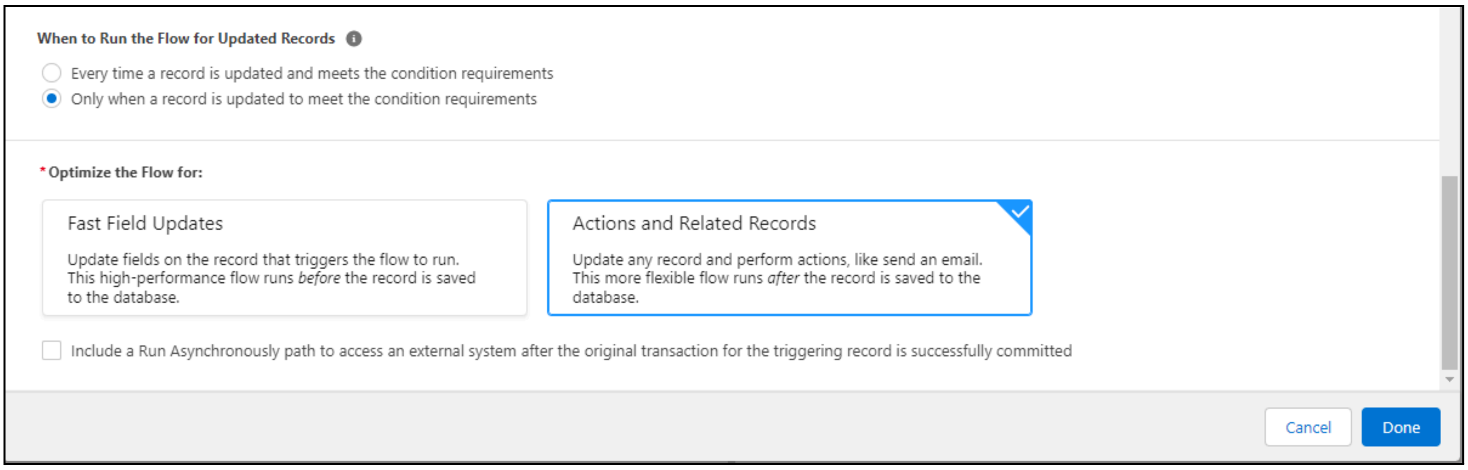
Condition Requirements: All Conditions Are Met (AND)

Field	Operator	Value
MobilePhone	Is Null	False

When to Run the Flow for Updated Records

- ○ Every time a record is updated and meets the condition requirements
- ● Only when a record is updated to meet the condition requirements

Fig: 4.1 Set entry conditions



When to Run the Flow for Updated Records

( ) Every time a record is updated and meets the condition requirements
(●) Only when a record is updated to meet the condition requirements

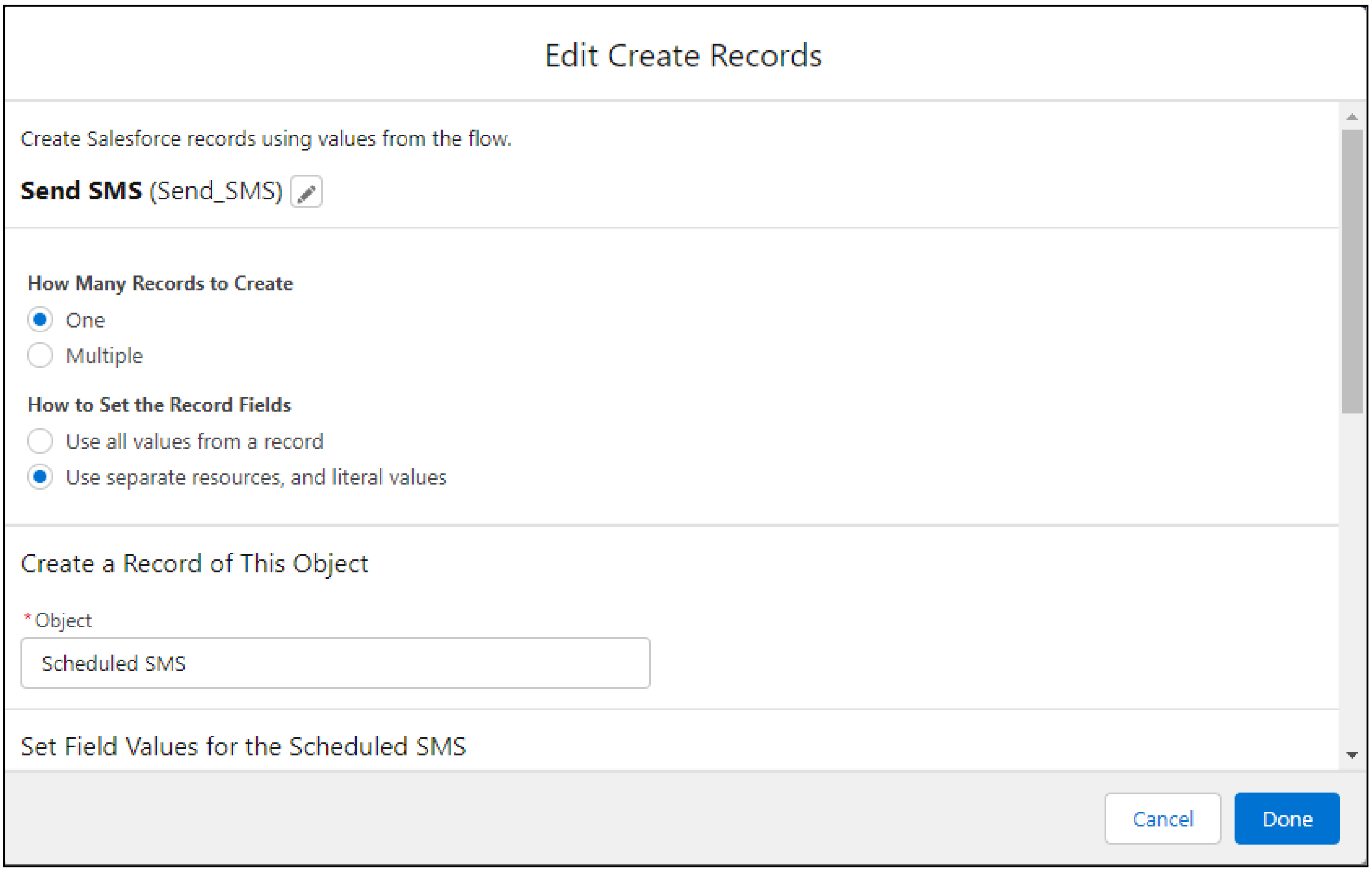
* Optimize the Flow for:

Fast Field Updates
Update fields on the record that triggers the flow to run. This high-performance flow runs *before* the record is saved to the database.

[✓ Actions and Related Records]
Update any record and perform actions, like send an email. This more flexible flow runs *after* the record is saved to the database.

☐ Include a Run Asynchronously path to access an external system after the original transaction for the triggering record is successfully committed

Fig: 4.2 other setup for initial step in flow



Edit Create Records

Create Salesforce records using values from the flow.

Send SMS (Send_SMS)

How Many Records to Create
(●) One
( ) Multiple

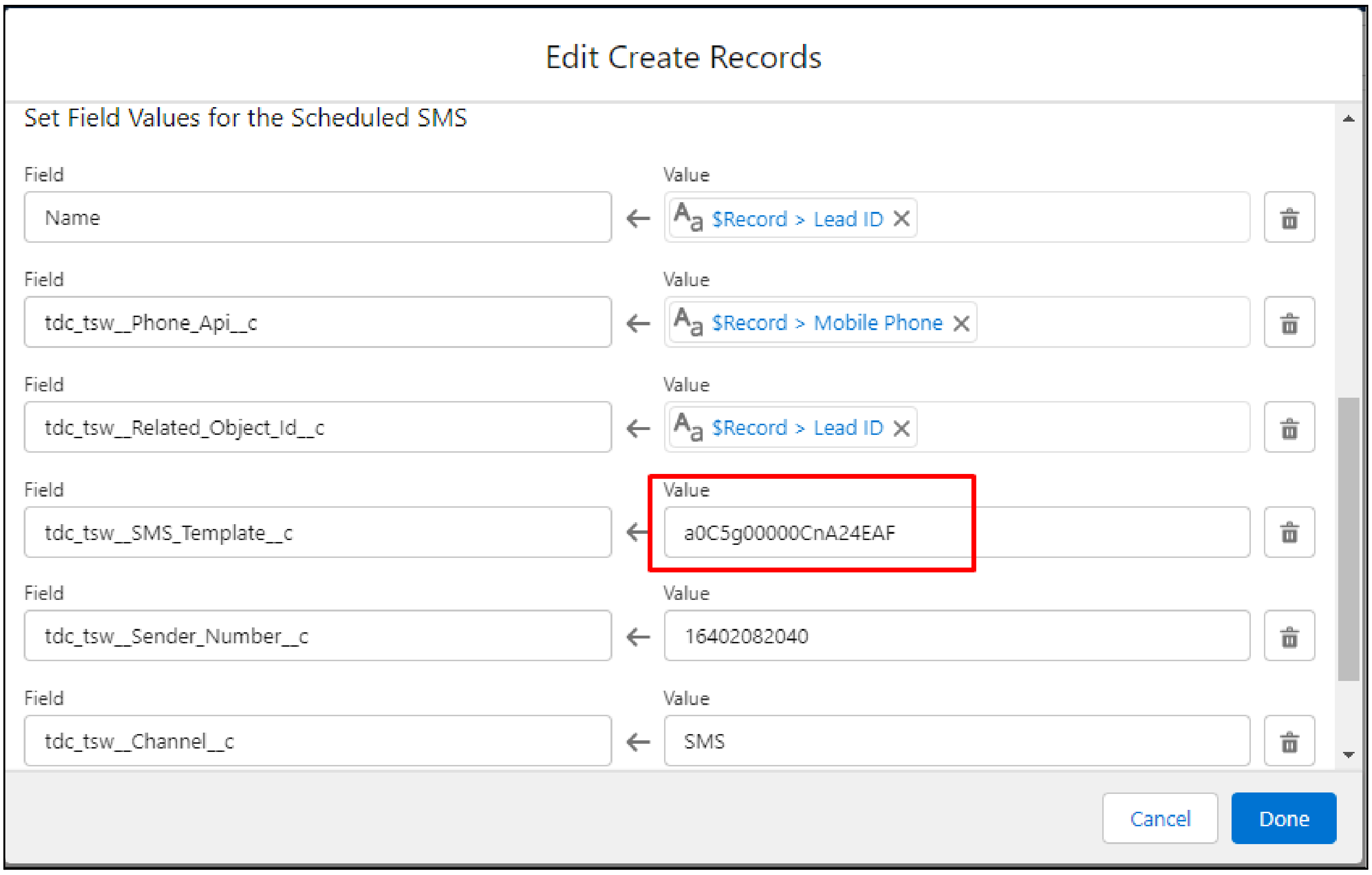
How to Set the Record Fields
( ) Use all values from a record
(●) Use separate resources, and literal values

Create a Record of This Object

* Object: Scheduled SMS

Set Field Values for the Scheduled SMS

Fig: 5 create records in flow


Edit Create Records

Set Field Values for the Scheduled SMS

Field	Value
Name	$Record > Lead ID
tdc_tsw__Phone_Api__c	$Record > Mobile Phone
tdc_tsw__Related_Object_Id__c	$Record > Lead ID
tdc_tsw__SMS_Template__c	a0C5g00000CnA24EAF
tdc_tsw__Sender_Number__c	16402082040
tdc_tsw__Channel__c	SMS

Fig: 5.1 The process of sending an SMS requires six essential fields. Additionally, there is an option to set the QUESTION field, which serves as the initial inquiry for the 360 SMS iText Object. When the QUESTION field is configured, it automatically triggers a Question/Answer iText sequence.

- **Scheduled SMS Name:** The ID field of the triggered Object/record must be provided. It should match with the SMS Template based on the Object to enable merging. For example, if you supply a Contact.Id, you should use a matching SMS Template based on the Contact Object.

- **Related Object Id:** Set the Related Object Id to either Lead.Id or Contact.Id. Additionally, you can set it to other Objects to gain visibility into the text conversations. For instance, setting it to an Account ID will attach the SMS History there. However, this approach will not set the SMS_History.Contact_Id.

- **Phone Api:** Provide the API Name of the phone field to which you want to send the message. It can be a string like "MobilePhone" or "Mobile_Phone c" (for a custom field), or you can reference the actual phone value from the record, such as Lead.MobileNumber. Alternatively, you can pull the value from the Incoming SMS SENDER_NUMBER, set it with a formula, or use User.MobileNumber for sending SMS to employees.

- When utilizing the Scheduled Time field to schedule the SMS, we highly recommend using the Field Name for the PhoneAPI value instead of the Field Reference. This is because the application will dynamically retrieve the phone value at the time of firing. If you explicitly set the value through Field Reference and the MobilePhone field is empty at that time, your SMS will not be triggered, even if you have set a value. Additionally, if the MobilePhone field changes between the scheduling of the message and its sending, the old value will be used. Therefore, to ensure the accurate and timely sending of SMS, it is advisable to use the Field Name for PhoneAPI value.
- **SMS Template:** Specify the ID of the template that you wish to use. You can obtain this ID from the template's URL.
- Alternatively, you can use a reference field like Contact.SMS_Template (if you have set up an SMS Template ID on your Contact). Please refer to the "Create a Master Send SMS Handler" section for more details.

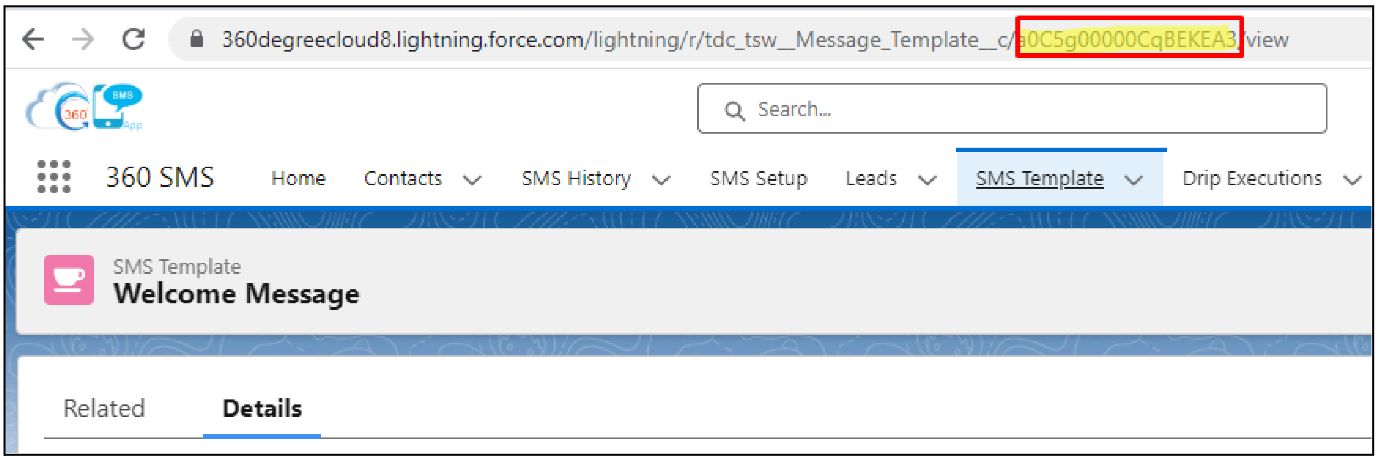


Fig:6 Template ID

Another way to create record trigger flow

The initial steps are the same as above to create a record trigger flow. However, in this method, we will not use "param"; instead, we can use toggles. We will add an action and, upon navigating to the action, perform a search. In the Apex application, the client must fill in the required details (as shown in figure). Moreover, they can select options from the toggle based on their specific requirements.

Click to “Add” button and select “Action” in flow

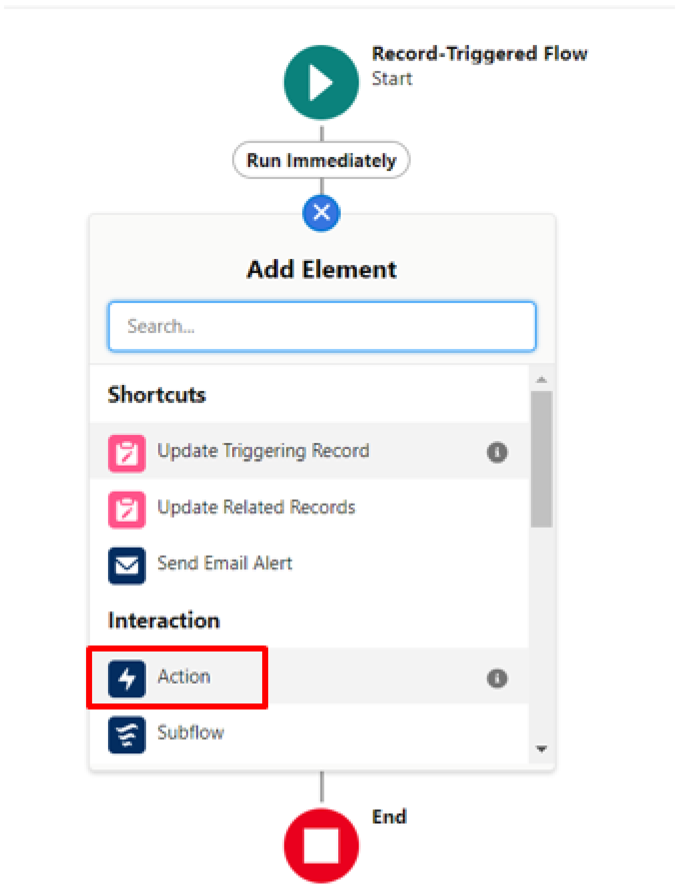
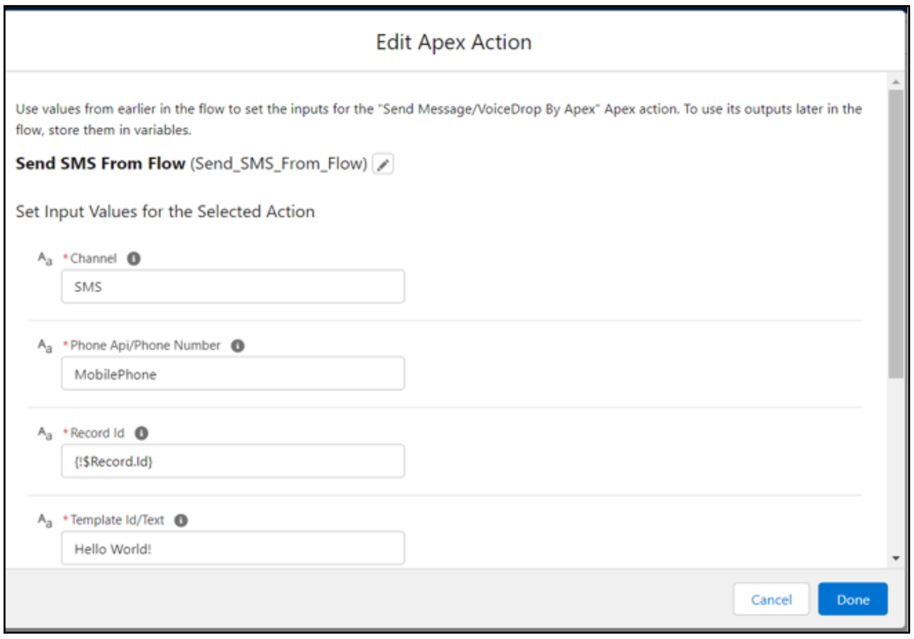


Fig:7 add “Action” in flow

Fill the required details (as shown in figure):



Edit Apex Action

Use values from earlier in the flow to set the inputs for the "Send Message/VoiceDrop By Apex" Apex action. To use its outputs later in the flow, store them in variables.

Send SMS From Flow (Send_SMS_From_Flow)

Set Input Values for the Selected Action

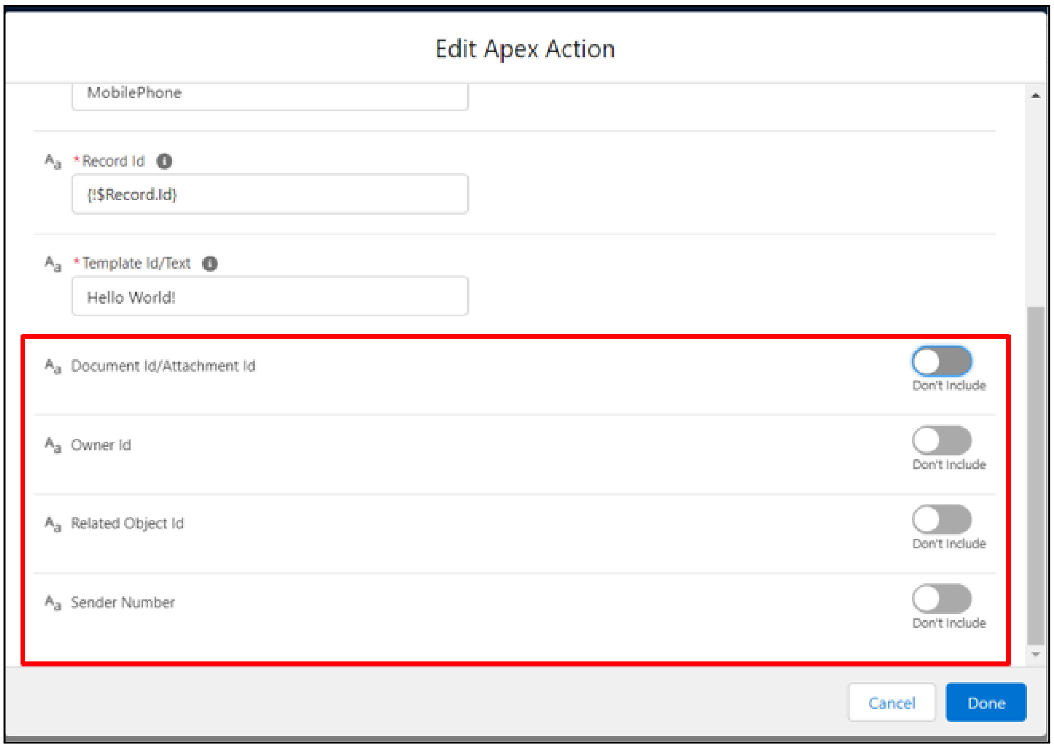
*Channel: SMS

*Phone Api/Phone Number: MobilePhone

*Record Id: {!$Record.Id}

*Template Id/Text: Hello World!

Fig: 8 Apex Action- fill the required details



Edit Apex Action

MobilePhone

*Record Id

{!$Record.Id}

*Template Id/Text

Hello World!

Document Id/Attachment Id — Don't Include

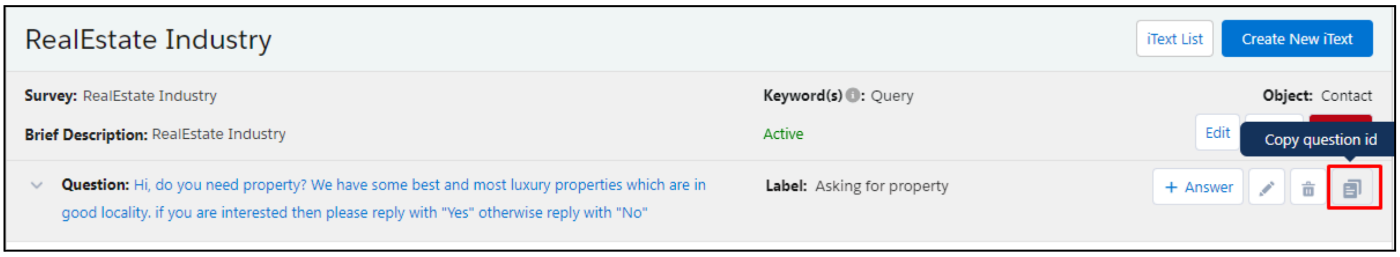
Owner Id — Don't Include

Related Object Id — Don't Include

Sender Number — Don't Include

Fig:9 Use toggles for more customisation

- **Question**: Instead of setting an SMS Template, you have the option to set the Question field. The QuestionId corresponds to the first question of a 360 SMS iText. By doing so, the first question of the iText will be triggered, and subsequent responses will be automatically handled. We strongly encourage the utilization of the iText Object as it enables automatic handling of responses, which is particularly valuable since most outbound SMS messages are likely to generate a response.



RealEstate Industry

Survey: RealEstate Industry

Brief Description: RealEstate Industry

Keyword(s): Query

Active

Object: Contact

Question: Hi, do you need property? We have some best and most luxury properties which are in good locality. if you are interested then please reply with "Yes" otherwise reply with "No"

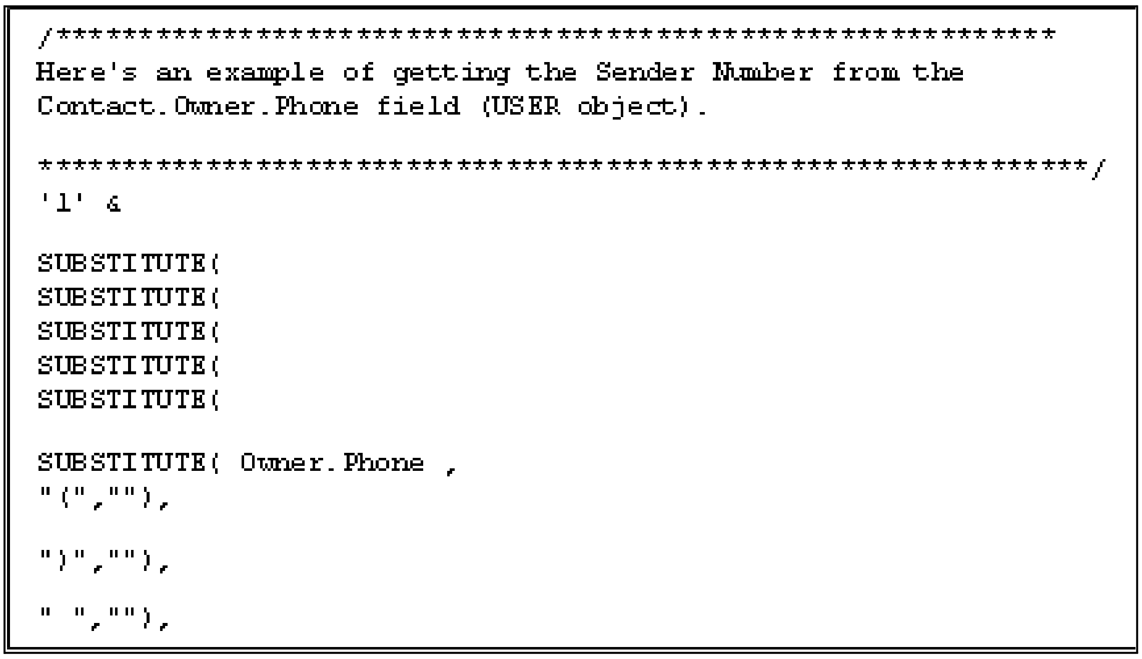
Label: Asking for property

Fig: 10 Obtain the question Id from the first question of a iText

- **Sender Number:** Specify the Sender Number, which is the number from which you are sending the message. You can also use a referenced field like

Lead.Owner.Phone to send from different salespeople, or a formula to assign different numbers based on geographies. If your organization has only one outbound number, this field is optional and can be left empty.

The sender number must always be in the format CountryCode+Number, without any formatting or special characters. For example, 17206050632. When dynamically setting the Sender Number, such as from Contact.Owner.Phone or from the Incoming SMS_History.To_Number, you can use the following valuable formula to remove all characters: [provide the formula].

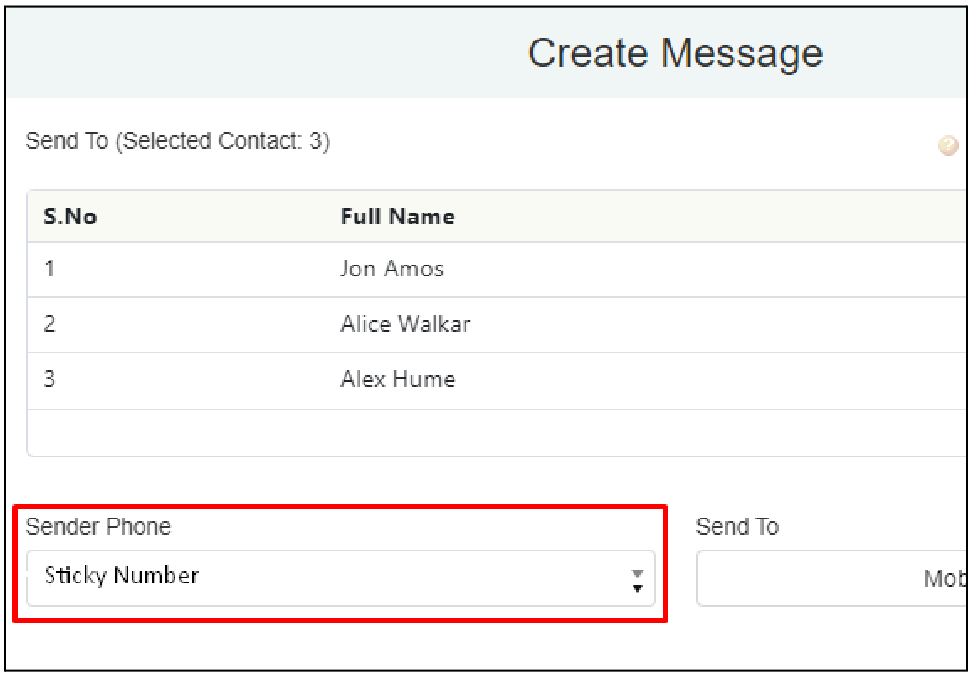


```
/***********************************************************
Here's an example of getting the Sender Number from the
Contact.Owner.Phone field (USER object).

*************************************************************/
'1' &

SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(

SUBSTITUTE( Owner.Phone ,
"(",""),

")",""),

" ",""),
```

Fig: 11 Useful formula for reducing a number to 17206050632 format

- We suggest utilizing the Sticky_Sender c field, which is a custom formula field recognized by the Send SMS buttons and Conversation View. It overrides the user's Default SMS Number with the formula value, ensuring that the number "sticks" to the record. This way, the customer always receives SMS from the same number.
- The formula field enables the definition of custom business logic for record-based sender numbers, guaranteeing consistent SMS delivery from the same number.
- This feature is particularly beneficial for geography-based solutions (such as differentiating between USA and UK numbers) or when each Record.Owner requires their own sender number.

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


Create Message

Send To (Selected Contact: 3)

S.No	Full Name
1	Jon Amos
2	Alice Walkar
3	Alex Hume

Sender Phone: Sticky Number

Send To: Mob

Fig: 12 Sticky sender

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

a. When utilizing this functionality, it is essential to configure the Lookup field of the relevant Object under which the Related List should be displayed, such as Contact or Lead. It's important to note that setting the Related Object field is distinct from setting ContactId or LeadId, as this field serves a different purpose.

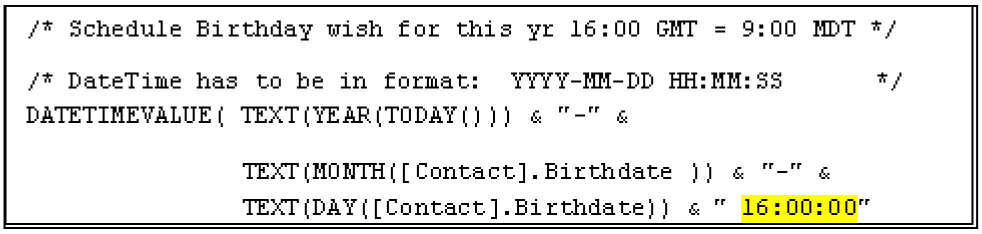
b. Typically, you would employ formulas like the following examples. Keep in mind that Salesforce date math operates in DAYS, so if you need to work with minutes and hours, you may need to look up the appropriate syntax.

i. NOW() + 1 /* Represents the same time as today plus 1 day, i.e., tomorrow /

ii. NOW() + (1/24) / Denotes 1 hour from the current time /

iii. NOW() + (10/60/24) / Indicates 10 minutes from the current time */

iv. Specifying a specific time of day can be challenging due to Salesforce storing time in Greenwich Mean Time (GMT). To account for this, you must adjust the time accordingly. For instance, if you want to schedule a Happy Birthday SMS to go out at 9:00 am Mountain Time, which is -7 hrs from GMT, you would set the time as 16:00:00 GMT.



```
/* Schedule Birthday wish for this yr 16:00 GMT = 9:00 MDT */

/* DateTime has to be in format:  YYYY-MM-DD HH:MM:SS       */
DATETIMEVALUE( TEXT(YEAR(TODAY())) & "-" &

               TEXT(MONTH([Contact].Birthdate )) & "-" &
               TEXT(DAY([Contact].Birthdate)) & " 16:00:00"
```

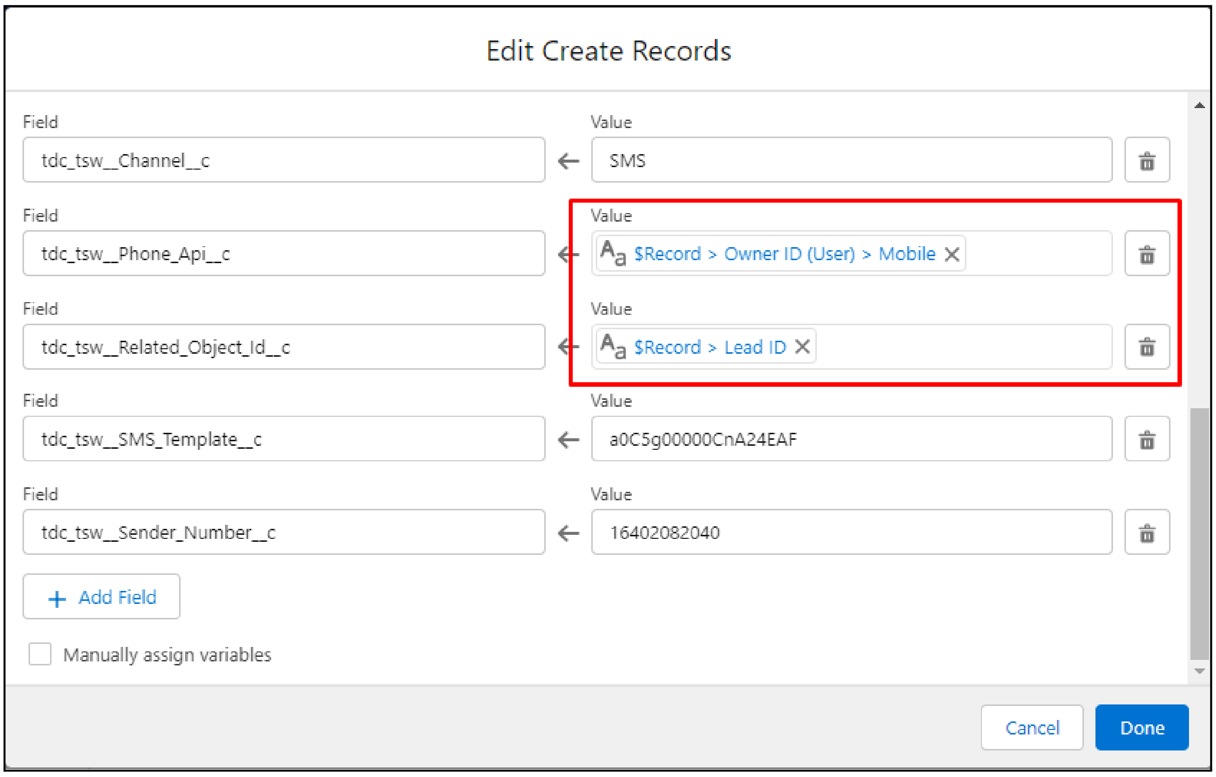
Relating Outbound SMS to an Alternate Object

The Related Object Id serves a specific purpose for Method #1, providing interesting applications when there's a need to associate a message with an Object other than its point of initiation or its template's source. This functionality is particularly valuable when handling replies that return to the related Object. Several use cases demonstrate its usefulness:

It allows linking a message to the parent CONTACT Object when the message originates from a Salesforce EVENT or another child Object, like event reminders.
Messages triggered from an Opportunity can be linked to a primary Contact or Account.
A significant application is directing messages to internal users. In this scenario, the Related Object Id is used to link the message to the USER Object rather than the main Object. This ensures that internal notifications do not appear in the Contact/Lead SMS History and avoid confusion as if the message was sent to the customer.
For a technique called "Roll-up SMS Conversations to Parent Objects," refer to the related section. This technique enables SMS to be sent from the child Object but also includes a flow to set the parent Contact and/or Account Object IDs.

Here's an illustrative example: when a new Lead is created from the Web Site, it triggers an internal user notification SMS. The Phone API is pulled from **Lead.Owner.User.MobilePhone**, and the Related Object Id is set to **Lead.OwnerId**. This approach allows using a template based on the Lead Object, providing Lead.Owner with the lead details while preventing the SMS History from appearing under the Lead itself.

It's important to note that the SMS Template's Object must match the Object of the ID provided in the Scheduled SMS Name field, and it is unrelated to the Related Object Id.



Edit Create Records

Field	Value
tdc_tsw__Channel__c	SMS
tdc_tsw__Phone_Api__c	$Record > Owner ID (User) > Mobile
tdc_tsw__Related_Object_Id__c	$Record > Lead ID
tdc_tsw__SMS_Template__c	a0C5g00000CnA24EAF
tdc_tsw__Sender_Number__c	16402082040

Fig: 13 A standard internal alert is sent to an employee via SMS, but the trigger originates from the creation of a Lead.

Method #2 – Apex Class

360 SMS offers two Apex classes that can be utilized either in Flows or trigger code. One class is dedicated to sending regular SMS, while the other is designed for sending MMS, allowing you to include a picture or file as a parameter.

To use these Apex classes, you pass parameters in a comma-separated string to the "param" field of the class, which offers the convenience of copying and pasting into complex workflows. An advantage of this method is that Salesforce permits comments in formula fields, using the /* some comment */ syntax, so it is strongly recommended to include comments in your formulas.

For the regular Send SMS From Flows Apex Class, the string of parameters is defined as follows:

Param1: The Id of the primary Object from which you are triggering the SMS. This Id must match your Template Object, and it will be the main Object linked to the outbound SMS.

Param2: The API name of the phone field for that Object, or it can be any value that results in a valid phone number.

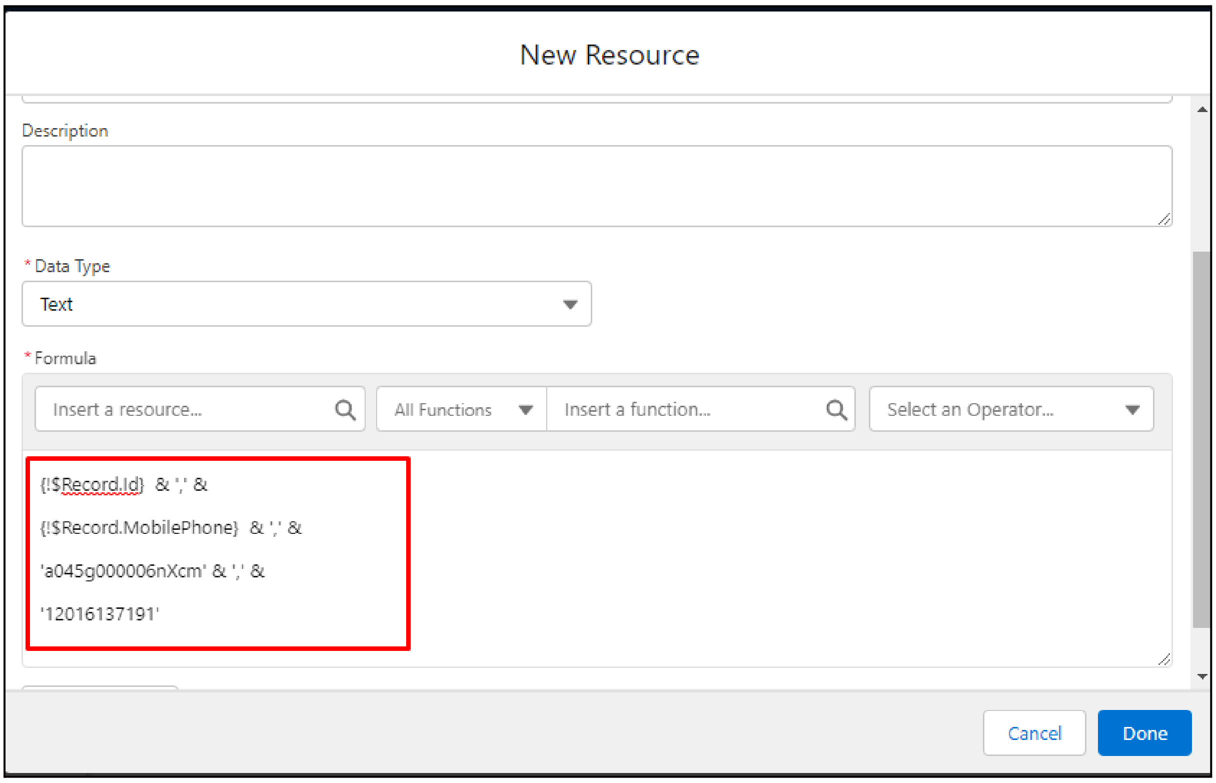
Param3: This parameter can accept one of the following options:

SMS TemplateId: Use a Template Id obtained from the URL of an SMS Template. The Object of the Template must match the Object defined in Param1.

QuestionId: Utilize a Question Id, typically associated with the first Question of an iText. This will trigger iText question #1, and all responses will be automatically handled.

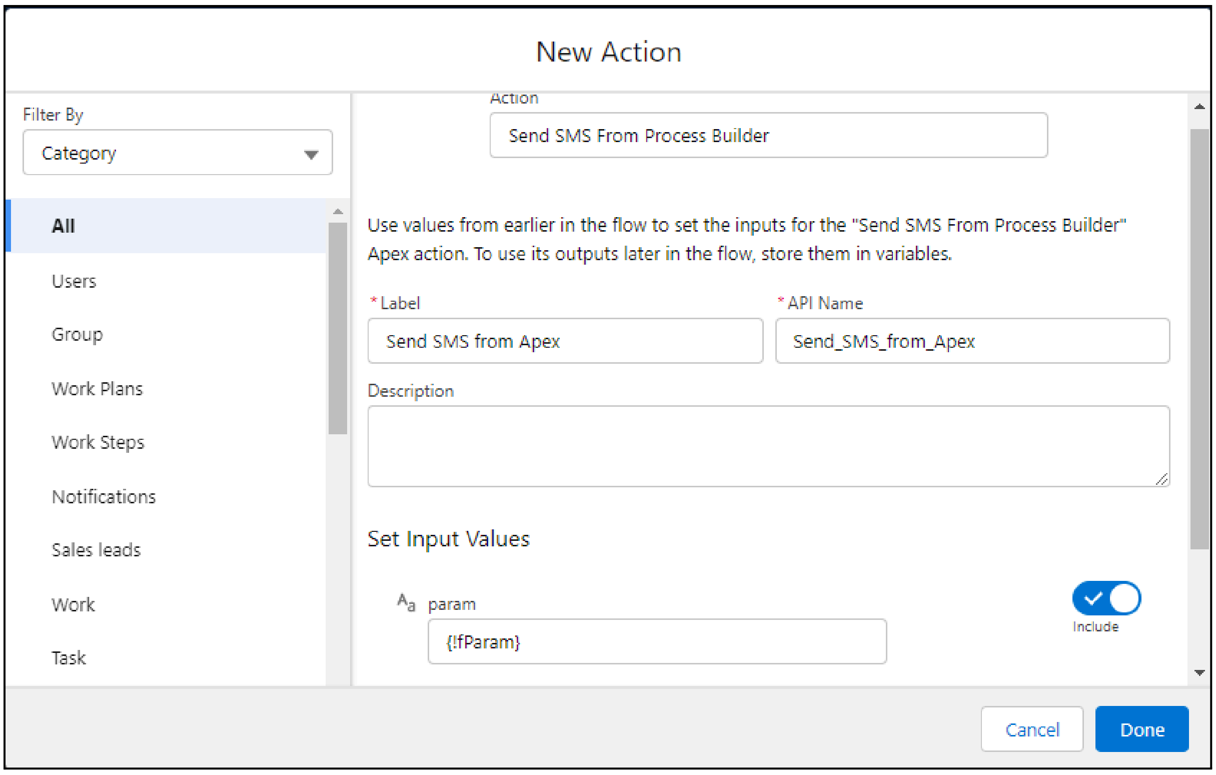
String: If you prefer not to use templates and require a simple SMS message, you can pass a string in this parameter. The string also supports merge tags, allowing you to personalize the message. For example: 'Okay {Contact.firstname} – this is a message without a template'.

Param4: The outbound phone number. If left blank, the default phone number for the organization will be used, or the first phone number found in the 360 SMS User Configuration tables for the current user will be utilized.



```
{!$Record.Id}  & ',' &
{!$Record.MobilePhone}  & ',' &
'a045g000006nXcm' & ',' &
'12016137191'
```

FIg:14 Send SMS through Apex class


New Action

Filter By: Category

Action: Send SMS From Process Builder

Use values from earlier in the flow to set the inputs for the "Send SMS From Process Builder" Apex action. To use its outputs later in the flow, store them in variables.

*Label: Send SMS from Apex

*API Name: Send_SMS_from_Apex

Description:

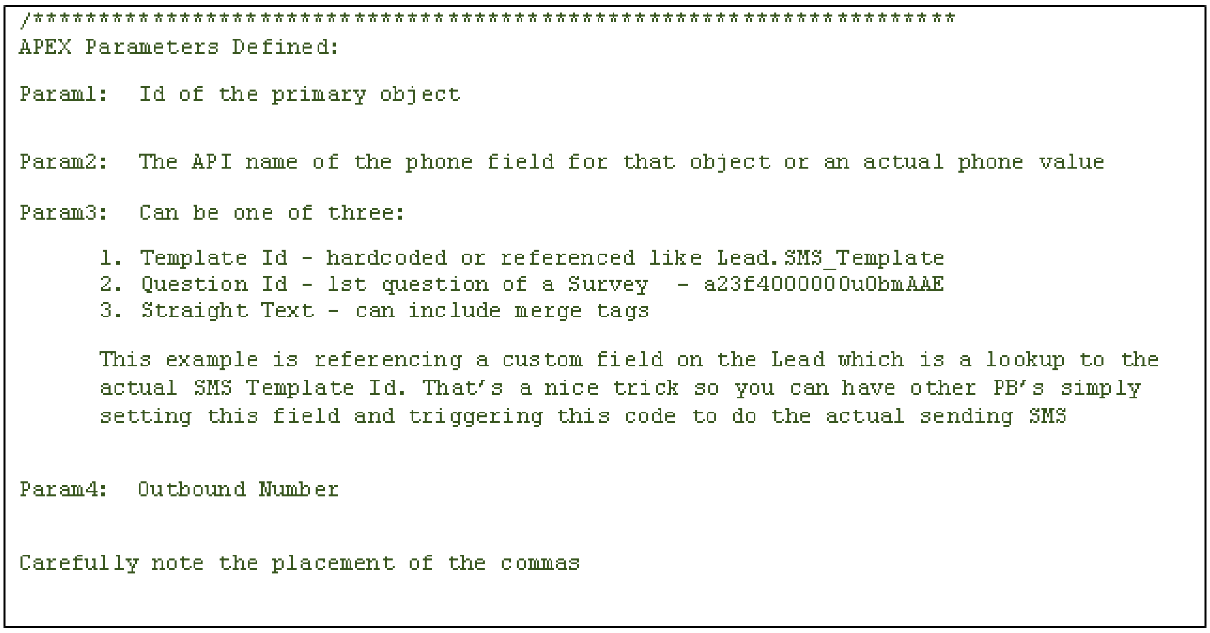
Set Input Values

param: {!fParam}

FIg:14.1 Send SMS through Apex class

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

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

APEX Parameters Defined:

Param1:   Id of the primary object


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

Param3:   Can be one of three:

      1. Template Id - hardcoded or referenced like Lead.SMS_Template
      2. Question Id - 1st question of a Survey  - a23f4000000u0bmAAE
      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
```



```
/*********************************************************************
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  - a23f4000000u0bmAAE
      3. Straight Text - can include merge tags

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


Param4:  Outbound Number


Carefully note the placement of the commas
```

Method #3 – Apex Triggers

For more advanced developers who favor using APEX Triggers instead of Flows, you can easily invoke the APEX class named tdc_tsw.GlobalSMSSender.sendSmsWithPhoneAPiAndTemplateId from your code.

This class requires three parameters:

Mobile Phone Field: Refers to the field where the contact's phone number is stored, for instance, objContact.MobilePhone.

Template Id: Corresponds to the TemplateId mentioned in the methods described earlier, like 'a022800000Pj0Ia'.

Object Id: Represents the ID of the record to which the SMS is being sent, such as objContact.id.

Below is a sample trigger for the CONTACT Object:

Trigger (for Contact Object)

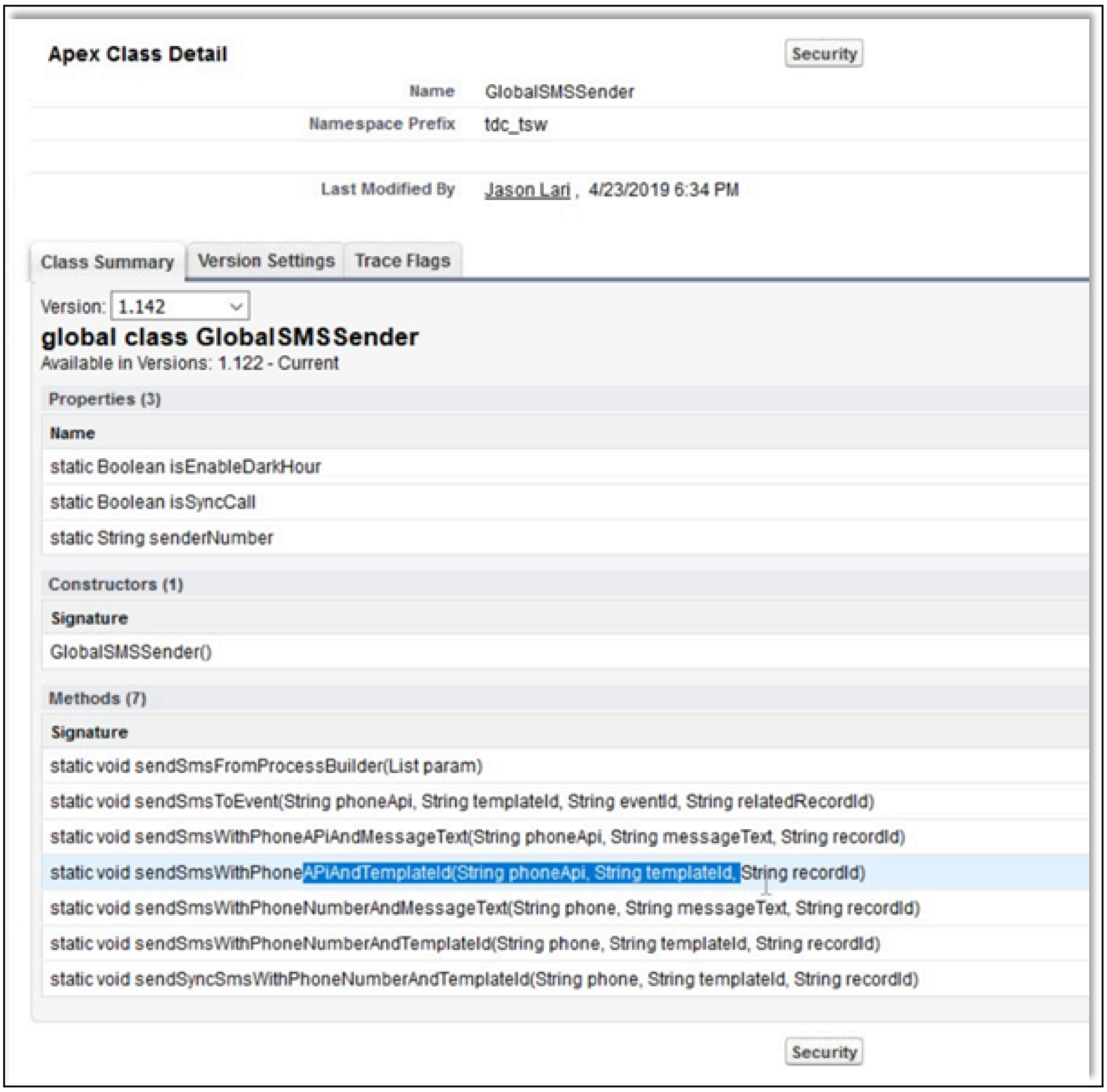
trigger ContactTrigger on Contact (after insert) { if(trigger.isAfter && trigger.isInsert) {

```
                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

Fig: 15 Additional APEX methods that be called via APEX code

Send MMS Via Flows:

MMS, short for Multimedia Messaging Service, refers to the transmission or reception of pictures within a text message. The set of parameters for MMS is explained below:

Param1: This is the ID of the primary Object that triggers the MMS. It must match your Template Object and will be the main Object associated with the outgoing SMS.

Param2: The API name of the phone field for the aforementioned Object.

Param3: This parameter offers three options:

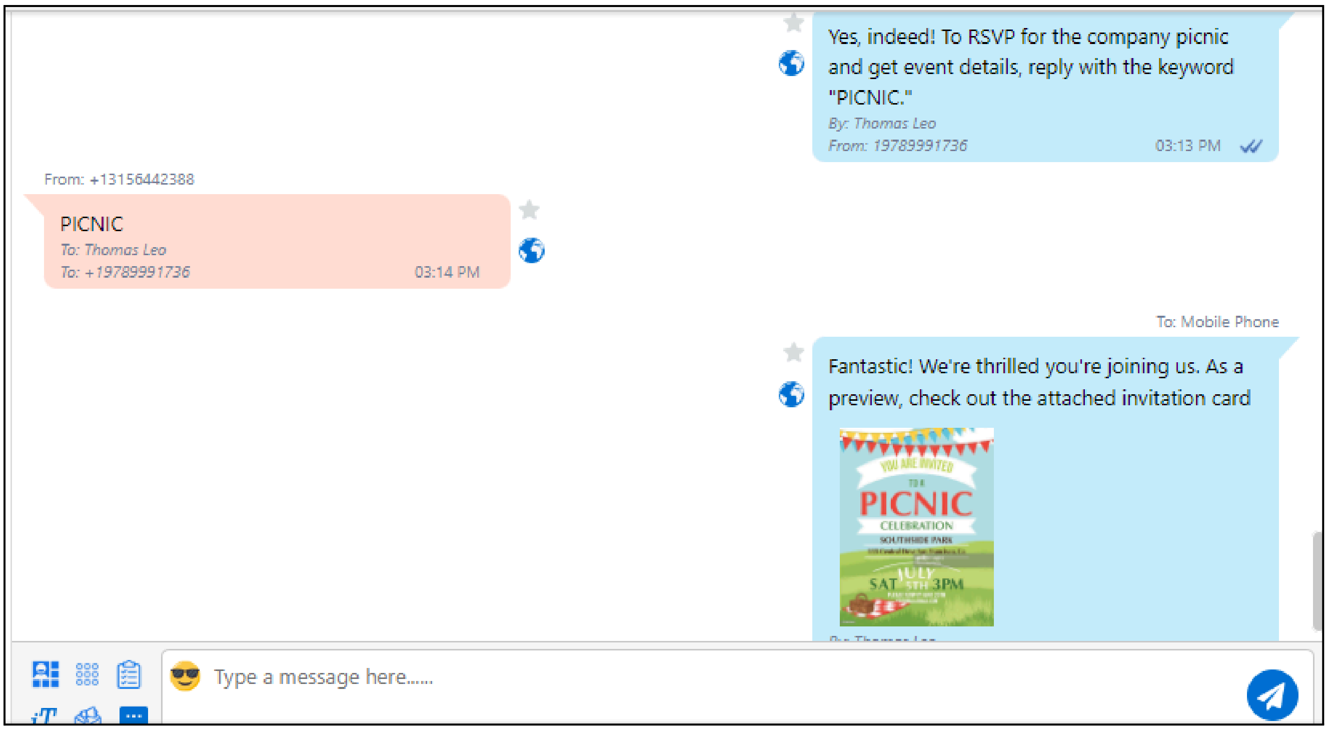
SMS TemplateId: Utilize a Template Id obtained from the URL of an SMS Template. The Object of the Template must match the Object defined in Param1.

QuestionId: Use a Question Id, typically linked to the first Question of an iText. This will trigger iText question #1, and all responses will be automatically managed.

String: If you prefer not to use templates and require a straightforward SMS message, you can pass a string in this parameter. The string also supports merge tags, allowing you to personalize the message. For example: 'Okay {Contact.firstname} – this is a message without a template.'

Param4: An optional Document ID that represents the picture or file to be sent along with the MMS.

Param5: An optional originating phone number. If left blank, it will use the default phone number for the organization or the first phone number found in the 360 SMS User Configuration tables for the current user.



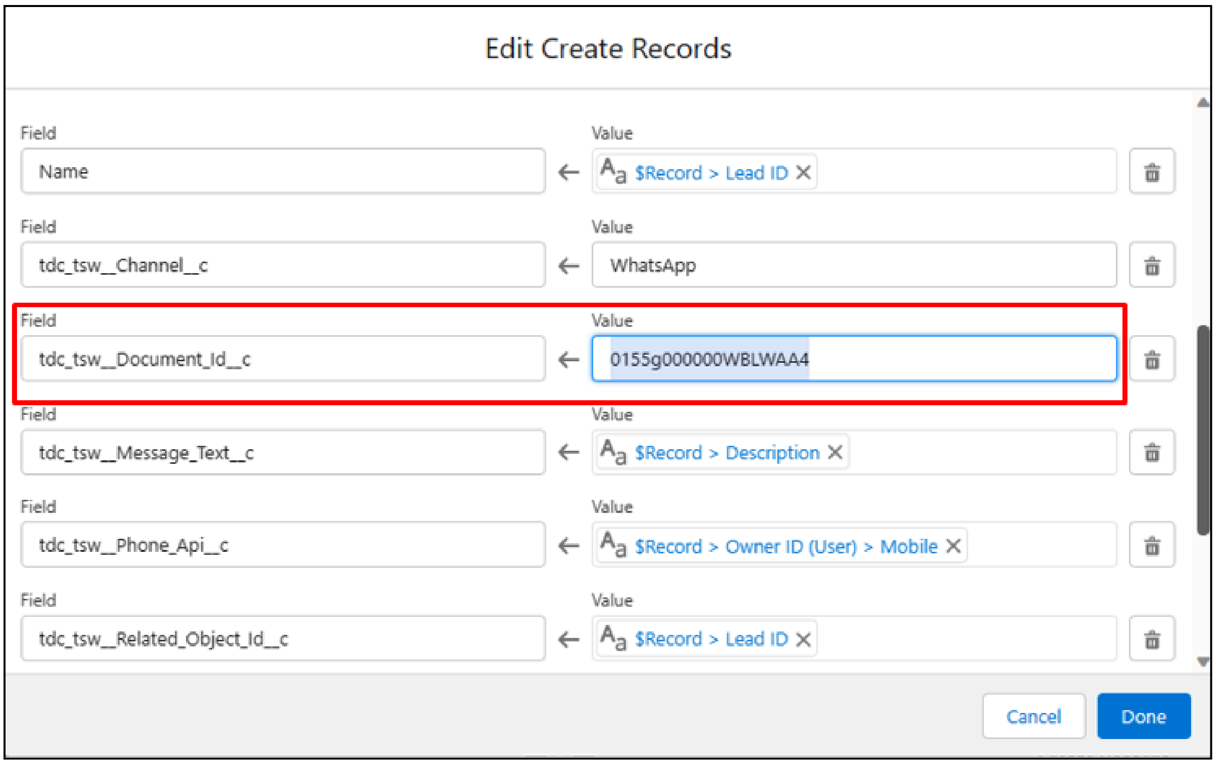
Yes, indeed! To RSVP for the company picnic and get event details, reply with the keyword "PICNIC."
By: Thomas Leo
From: 19789991736 03:13 PM

From: +13156442388
PICNIC
To: Thomas Leo
To: +19789991736 03:14 PM

To: Mobile Phone
Fantastic! We're thrilled you're joining us. As a preview, check out the attached invitation card

Fig: 16 MMS functionality using the keyword "PICNIC," which sends a picture of picnic invitation card

Figure 13.1 Also demonstrates a completely dynamic solution where the picture is derived by navigating to a custom field on the User record via SMS_History.Owner and the outbound phone number is also gathered from the User record. Most of the time you will be dynamically setting the Pictures and Outbound Number.



Edit Create Records

Field	Value
Name	$Record > Lead ID
tdc_tsw__Channel__c	WhatsApp
tdc_tsw__Document_Id__c	0155g000000WBLWAA4
tdc_tsw__Message_Text__c	$Record > Description
tdc_tsw__Phone_Api__c	$Record > Owner ID (User) > Mobile
tdc_tsw__Related_Object_Id__c	$Record > Lead ID

Fig: 17 In order to send an MMS through Salesforce flows, it is usually necessary to have a document ID, and the document itself must be stored in the Document Object within Salesforce.

Dynamic Sender Number

Regardless of the chosen method, developers often aim to set the outbound Sender Number dynamically to ensure a consistent customer experience with a single phone number. This is particularly relevant for organizations using multiple sender numbers.

The Sender Number must always adhere to the format CountryCode+Number, without any formatting or special characters, for example, 17206050632. To achieve this, when dynamically setting the Sender Number, like extracting it from Contact.Owner.Phone or 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 , "(",""),

")",""),

" ",""),
"-",""),
"+",""),
".","")
```

Fig: 18 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.

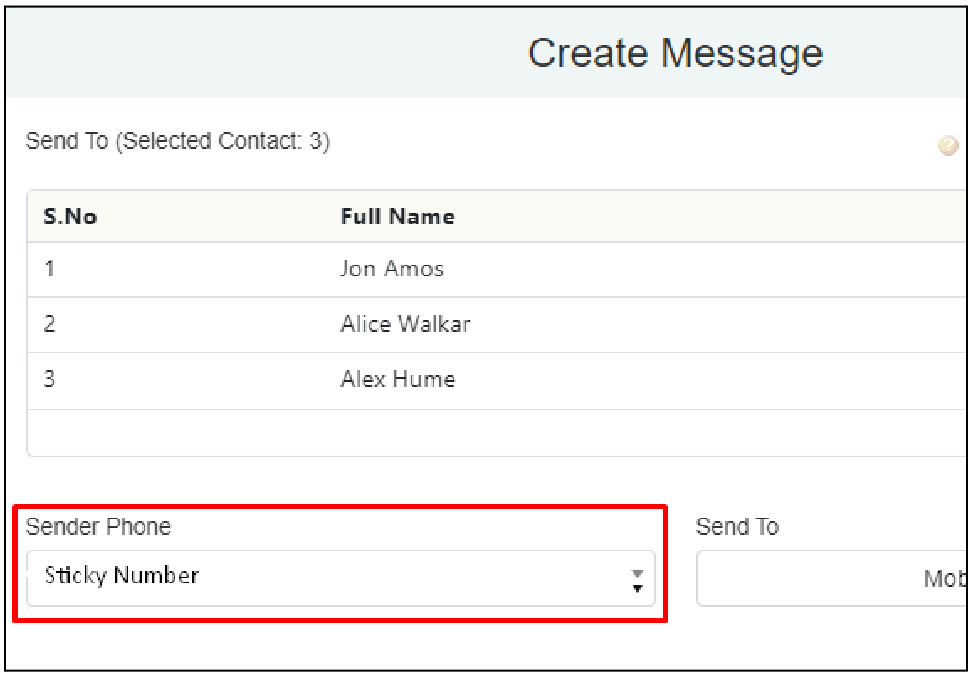
4. **Automatically replying to an incoming SMS using the number which the customer wrote to**, the To_Number for an SMS_History of type = Incoming (API name: `tdc_tsw_ToNumber_c`)

a. Note that the 360SMS iText tool does this automatically – always responding to the original number that the customer wrote to.

Record based Sender Number (Sticky Sender)

360 SMS provides the flexibility to create a custom field named Sticky_Sender c (the label can be customized according to preference). This custom formula field can hold any desired business logic, such as Geography-Based numbers, Marketing Campaign Based Numbers, Record.Owner based numbers, or even the Last SMS number used for the contact. The purpose is to allow the definition of various business rules as needed.

Once this field is created, all interface elements of the platform, including Buttons and Conversation View, will recognize Sticky_Sender and override the current user's Default SMS Number. Instead, they will offer this custom field as the default Sender Number. For example, a marketing user may wish to select a large list of contacts for batch SMS but wants the Sender Number to be the one that the customer has previously received messages from, such as the [Record].Owner's sender number. This level of customization empowers users to tailor their messaging approach to suit specific scenarios.


Create Message

Send To (Selected Contact: 3)

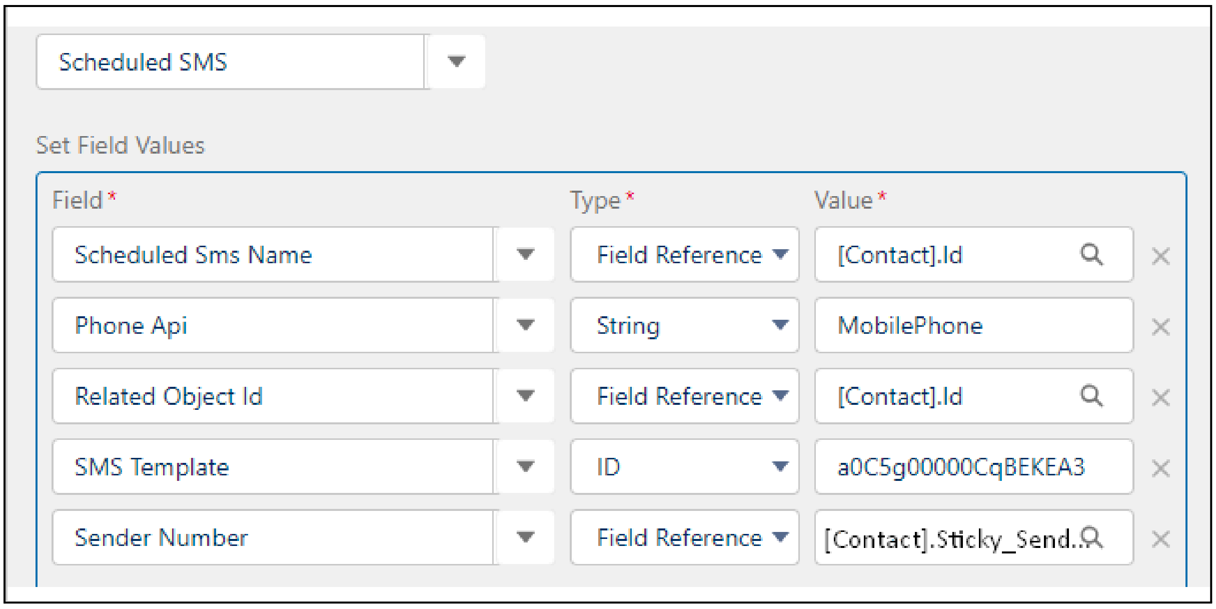
S.No	Full Name
1	Jon Amos
2	Alice Walkar
3	Alex Hume

Sender Phone: Sticky Number

Send To: Mob

Fig: 19 Once a sticky sender formula field is defined, it will automatically serve as the default Sender Phone for batch, single, and conversation 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.



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	a0C5g00000CqBEKEA3
Sender Number	Field Reference	[Contact].Sticky_Send.

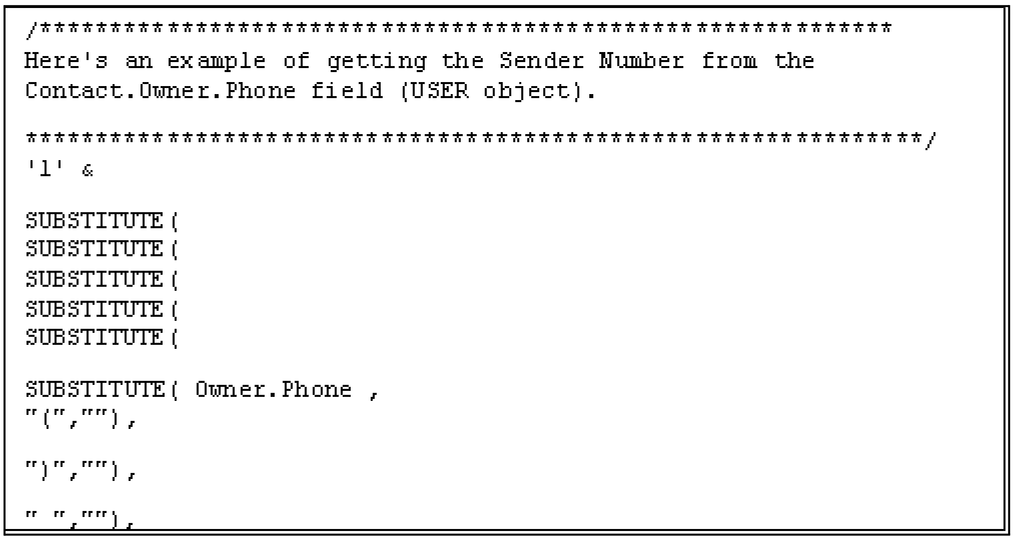
Fig: 20 Employ the sticky sender field to maintain a consistent Sender Number and centralize your business logic within the formula field.

Dynamic Sender Number – Record Owner

Many organizations use different numbers for each user, mainly for voice call forwarding scenarios. The mapping of Users to Numbers is established in the SMS Setup à User Configuration, which, unfortunately, is not accessible via flows. However, you can easily customize your USER Object to enable flows to dynamically obtain the Outbound SMS Sender Number parameter available in both methods as an optional parameter.

To achieve this, create a custom text field named something like User.SMS_Number. Then, copy the number associated with each user into this field. Take caution when using the standard User.MobilePhone or Phone field, as Salesforce formats these numbers, such as (720)605-0632. The number should be completely unformatted, with the country code prefix, i.e., 17206050632.

Now, you can navigate to the User table, such as Lead.Owner/Contact.Owner, and retrieve the number from your custom field. Alternatively, define this logic in a Sticky_Sender c formula field as either Owner.SMS_Number c or use the provided formula below, which ensures the removal of various phone formatting characters.



```
/*************************************************************
Here's an example of getting the Sender Number from the
Contact.Owner.Phone field (USER object).

*************************************************************/
'1' &

SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(
SUBSTITUTE(

SUBSTITUTE( Owner.Phone ,
"(",""),

")",""),

" ",""),
```

Dynamic Sender Number – Geography or Business Rule Based

In larger organizations with diverse geographies or complex business rules, it is highly desirable, and sometimes even mandatory, to use specific sender numbers. When dealing with multiple countries, it becomes necessary to employ a sender number that aligns with the country of the recipient. Here are three solutions to address this:

Use the Sticky Sender custom formula field, as explained earlier.

Leverage the 360 SMS Co-Pilot feature, which employs its Area Code/Country Code picker to dynamically select the sender number. This can be done if you have purchased a matching Area Code or Country Code number in your pool. For more information on Co-Pilot, you can refer to the linked document: Batch Texting - Auto-Routing.

Utilize the formula editor in flows to incorporate your business logic for the Sender Number. However, it is advised to now place your logic in the Sticky Sender formula field rather than defining it in the flows' Dynamic Sender Number – Incoming SMS.

Another common scenario involves dynamically setting the SMS Number parameter based on the Incoming Message. This is frequently used when responding to Keywords. In this case, there is no need to look up the number from a user table; instead, you can retrieve it directly from the SMS_History.To_Number field (the number that the customer wrote to). Be cautious, though, as the value will have a "+" character in front of it, which is not valid for an outbound number. You must use the SUBSTITUTE function, as shown below, to remove the "+" character. 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 iText) 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' & ',' &

'a08f400000Dfo3fAA

B' & ',' &

SUBSTITUTE([tdc_tsw_Message_c].tdc_tsw_ToNumber_c , '+', '')
```

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

Similar to implementing a dynamic Sender Number, it is possible to create a custom field on the User record to store the Document Id of a previously uploaded picture.

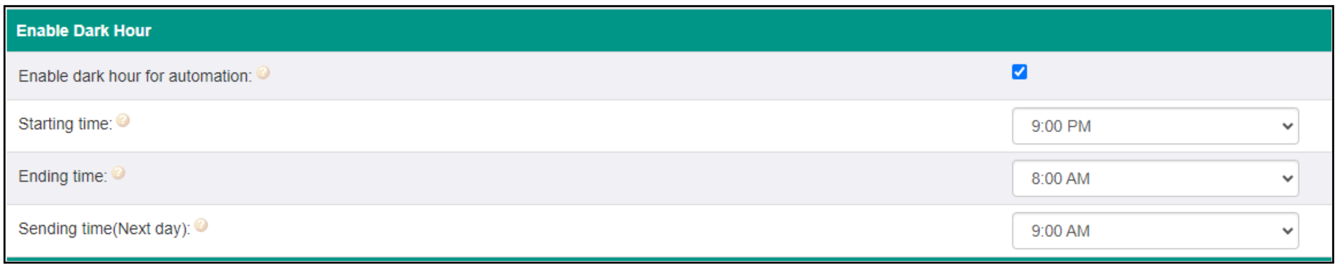
In this instance, let's consider an example where we uploaded a picture to the Salesforce Document Object. Afterward, we manually copied and pasted the actual ID of the picture into a custom field named User.Picture_Doc_Id. The Document Id was obtained from the URL when we accessed the picture.

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

```
/*********************************************************************** APEX
Parameters Defined:
Param1:   Id of the primary Object - pulled from
SMS_History.ContactId Param2:     The API name of the phone
field for that Object.
Param3:   Template Id: a08f400000DflORAAZ = Contact - Event
Reminder Param4:       Document Id pulled from the
Owner.Picture_Doc_Id field

Param5:   Optional Outgoing Phone # - pulled from Contact.Owner --> User.SMS_Number
(custom)



Carefully note the placement of the commas

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



[tdc_tsw_Message_c].tdc_tsw_Contact_c   & ',' &
'MobilePhone' & ',' &

'a08f400000DflORAAZ' & ',' &

[tdc_tsw_Message_c].tdc_tsw_Contact_c.Owner.Picture_Doc_Id_c & ',' & [tdc_tsw
Message_c].tdc_tsw_Contact_c.Owner.SMS_Number_c
```

Dark Hours

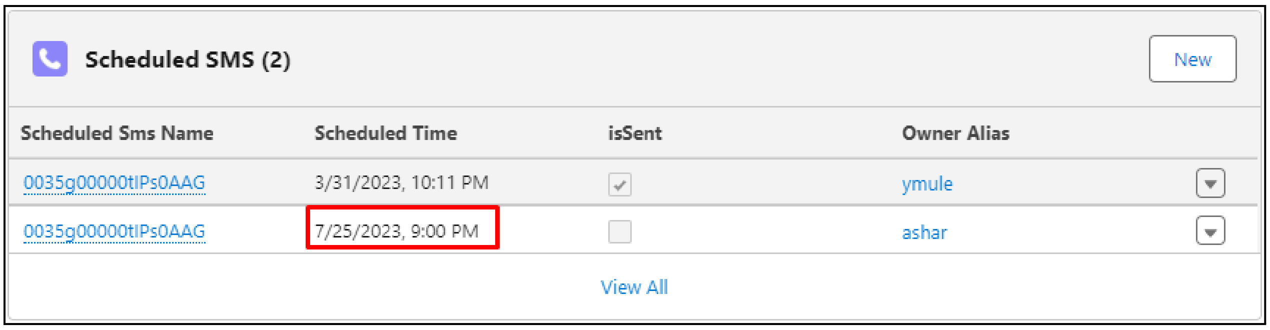
360SMS includes a feature called Dark Hours, which, when activated, postpones the firing of any Triggered or Scheduled SMS messages until the designated "Sending time/Next Day." As shown in the screen capture below, any messages triggered between 4:00 PM and 6:00 AM will be held back and delivered at 6:00 AM on the following day.



Enable Dark Hour	
Enable dark hour for automation:	☑
Starting time:	9:00 PM
Ending time:	8:00 AM
Sending time(Next day):	9:00 AM

Fig: 21 With Dark Hours enabled, any triggered messages will be scheduled to be sent on the following day at the specified time

In technical terms, if a process initiates the sending of an SMS during the dark hour period, it will generate a Scheduled SMS related list record, as depicted below, with the Schedule Date/Time feature. Subsequently, the Scheduled SMS will be executed on the following day at the specified start time.


Scheduled SMS (2)

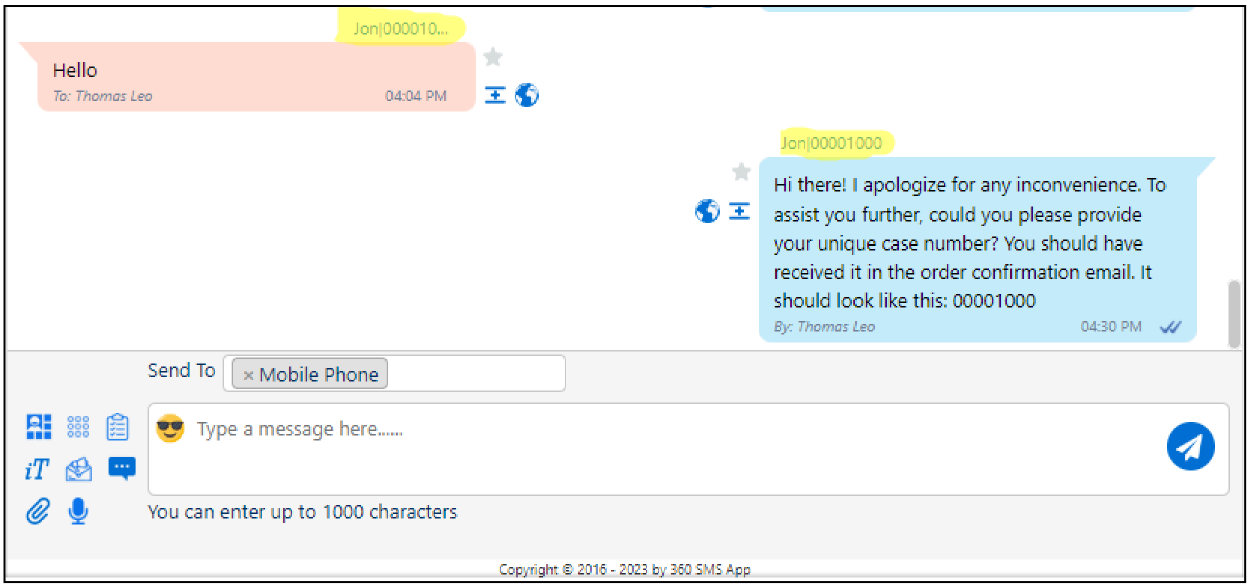
Scheduled Sms Name	Scheduled Time	isSent	Owner Alias
0035g00000tIPs0AAG	3/31/2023, 10:11 PM	✓	ymule
0035g00000tIPs0AAG	7/25/2023, 9:00 PM		ashar

Fig: 22 - *When Triggered SMS occurs during dark hours, it generates a Scheduled SMS record, and the Scheduled Time will be set to the sending time on the next day.*

Roll-up SMS Conversations to Parent Objects

A recommended best practice in CRM involves ensuring that when sending text messages, whether done manually or through automation, from a child Object, the SMS should also be visible on the parent CONTACT Object. Similarly, if texting is initiated from the Contact, the SMS should be rolled-up to the Account, allowing all conversations with various contacts to be accessible on the Account page.

For the CASE Object, any SMS linked to it is automatically rolled-up to the parent Contact. This approach ensures that when viewing the Contact record, all conversations are presented in context, relating to both the Contact and the Case. However, for other Objects, the SMS History needs to be intercepted through Flows. Subsequently, an Immediate Action should be implemented to update the SMS History and perform the roll-up. This ensures that SMS conversations are properly tracked and displayed in the relevant context across different objects within the CRM.


Jon|000010...

Hello

To: Thomas Leo 04:04 PM

Jon|00001000

Hi there! I apologize for any inconvenience. To assist you further, could you please provide your unique case number? You should have received it in the order confirmation email. It should look like this: 00001000

By: Thomas Leo 04:30 PM

Send To × Mobile Phone

Type a message here......

You can enter up to 1000 characters

Copyright © 2016 - 2023 by 360 SMS App

Fig: 23 - Automatically consolidating text messages from various cases causes the conversation to be grouped together under the corresponding contact, as demonstrated with case 00001000. We aim to replicate this feature for other child objects, allowing similar functionality for those as well.

In the following example, we have a classic child Object called Job_Listing, which acts as a child to both a Job and a Contact. This situation typically occurs when a recruiting company offers a Job_Listing, and a Contact expresses potential interest in that particular job. Recruiting firms often send batch texts or trigger texts from the Job_Listing, and they want to view all the SMS messages not only in the Job's SMS Conversation View but also from the associated Contact records. They wish to see all the Job-related texts they have sent to the Contact, similar to how it works with Cases (as shown in Figure 15).

To achieve this, we utilize a ROLL UP approach. When an incoming SMS comes in for a Contact, and we need to update Contact fields accordingly, we detect that the SMS History's primary related Object is Job_Listing. As the Job_Listing is a child of both a Job and a Contact, we update the SMS History with these values. Specifically, we set SMS_History.ContactId = Job_Listing.ContactId and SMS_History.Job = Job_Listing.JobId.

It's important to note that the platform uses the primary Related Object's NAME field for the Sender Name (the link that appears above the message and is displayed in the Incoming Alert). To ensure better practice when texting from Child Objects, we modify the Sender Name with a Flow in this scenario. For instance, it reads "Jon | SF Administrator," where "Jon" is the Contact.Name, and "SF Administrator" is the Job.Name. This modification enhances the readability and context of the Sender Name in the messaging interface.

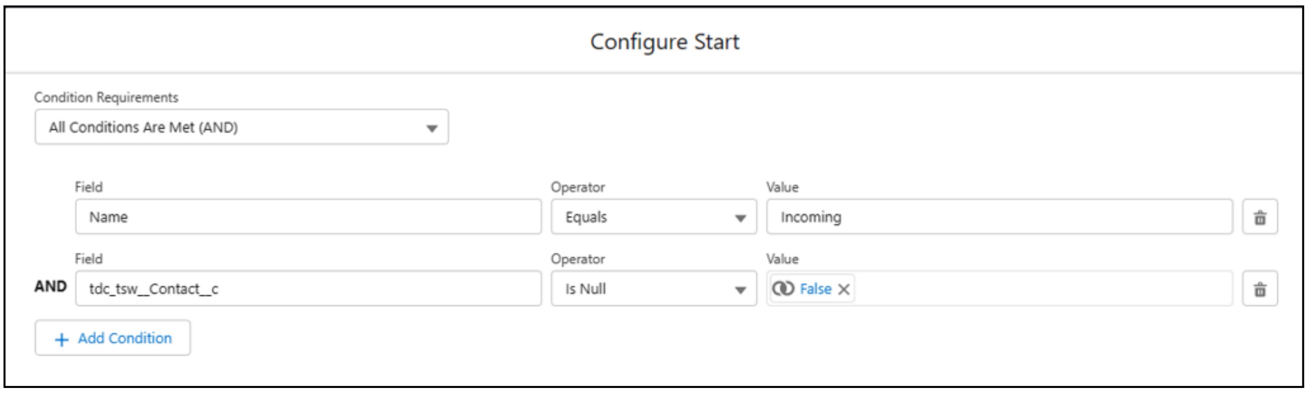
Create a Contact/Lead Last_SMS field

Frequently, customers want to utilize Salesforce List Views based on Contacts, Leads, or other custom Objects to determine which customers have or haven't received an SMS recently. However, due to Salesforce List Views' limitation of not allowing Cross-Object queries, and Salesforce Reports lacking a "Does Not Exist" clause, this presents a challenge.

To overcome this limitation in Salesforce, we suggest implementing a straightforward Salesforce Flow that updates a custom field, such as Contact.Last_SMS or Lead.Last_SMS, whenever an SMS History record is created. Optionally, you can differentiate between Incoming and Outgoing messages by using fields like Last_SMS_In and Last_SMS_Out, but usually, a single field is sufficient.

Create a Flow on the SMS History, like the "SMS History – Master Updater Process" mentioned in earlier sections. In this Flow, detect if the SMS History.Contact or SMS_History.Lead value is not null. If it is not null, then update the respective record's Last_SMS field with the SMS_History.Create_Date.

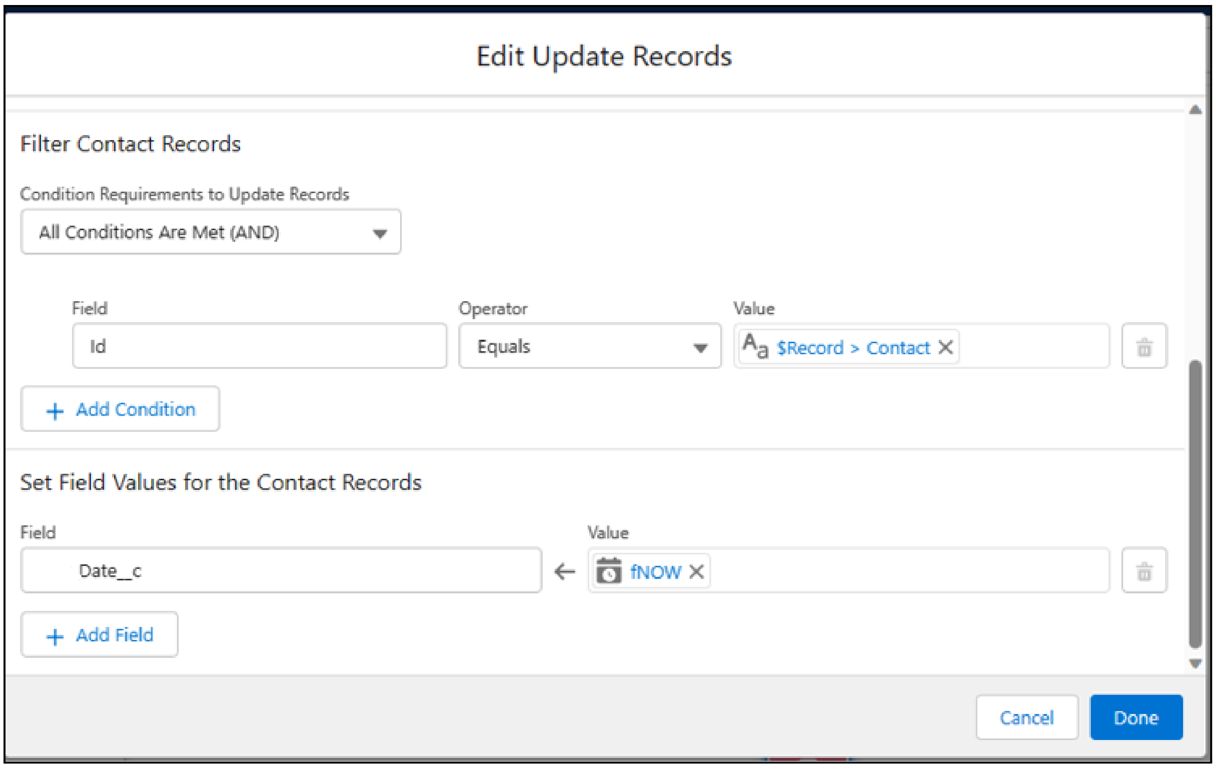
By implementing this Flow, you can now use the Last_SMS field in your Contact/Lead List View queries


Configure Start

Condition Requirements: All Conditions Are Met (AND)

	Field	Operator	Value
	Name	Equals	Incoming
AND	tdc_tsw__Contact__c	Is Null	False

Fig: 24 Create Contact/Lead Last SMS field


Edit Update Records

Filter Contact Records

Condition Requirements to Update Records: All Conditions Are Met (AND)

Field	Operator	Value
Id	Equals	$Record > Contact

Set Field Values for the Contact Records

Field		Value
Date__c	←	fNOW

Fig: 24.1 Create Contact/Lead Last SMS field

Triggered Texting to Internal Users

Certain customers wish to leverage the effectiveness of texting to replace their old email alert systems with Text Alerts for internal users. In the following scenario, we have a use case where a New Lead text alert is sent to the Lead.Owner's mobile phone, along with a hyperlink to quickly open the Salesforce record in Salesforce1.

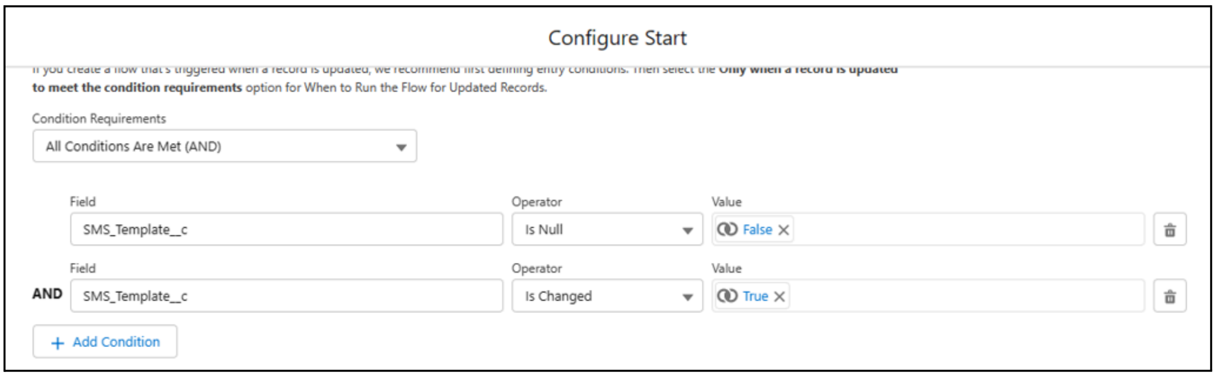
To achieve this, we trigger the message using the Lead.Id, enabling the use of a Lead-based template that merges lead details into the message. The key element is setting the Phone API field to extract the value from the Lead.Owner.User.MobilePhone, which ensures that the Lead details are sent to the Owner's personal cell phone.

An important consideration is to set the Related Object Id to Lead.OwnerId (representing a user id). This step ensures that the alert does not appear in the SMS History for the lead, avoiding any confusion and making it clear that this is not a message sent to the customer. By implementing these steps, users can effectively switch to text alerts and streamline their internal communication processes.

Create a Master Send SMS Handler

If you plan on using Method #1 for various triggers, you may consider adding an SMS_Template_Id lookup field to a Lead, Contact, or any custom Object. This approach allows you to have multiple Flows that trigger outbound messages, and all you need to do is update the Contact.SMS_Template with the desired template you want to send. This centralized approach eliminates the need to repeatedly create the same Send SMS action, regardless of whether you are using Method #1 or Method #2.

However, there will be instances when you may not want to use your Master Send SMS Handler. For example, this could be the case when the outbound number needs to come from a different number or when you need to trigger an MMS. In such situations, you can opt not to utilize the Master Send SMS Handler and tailor the messaging process according to specific requirements.



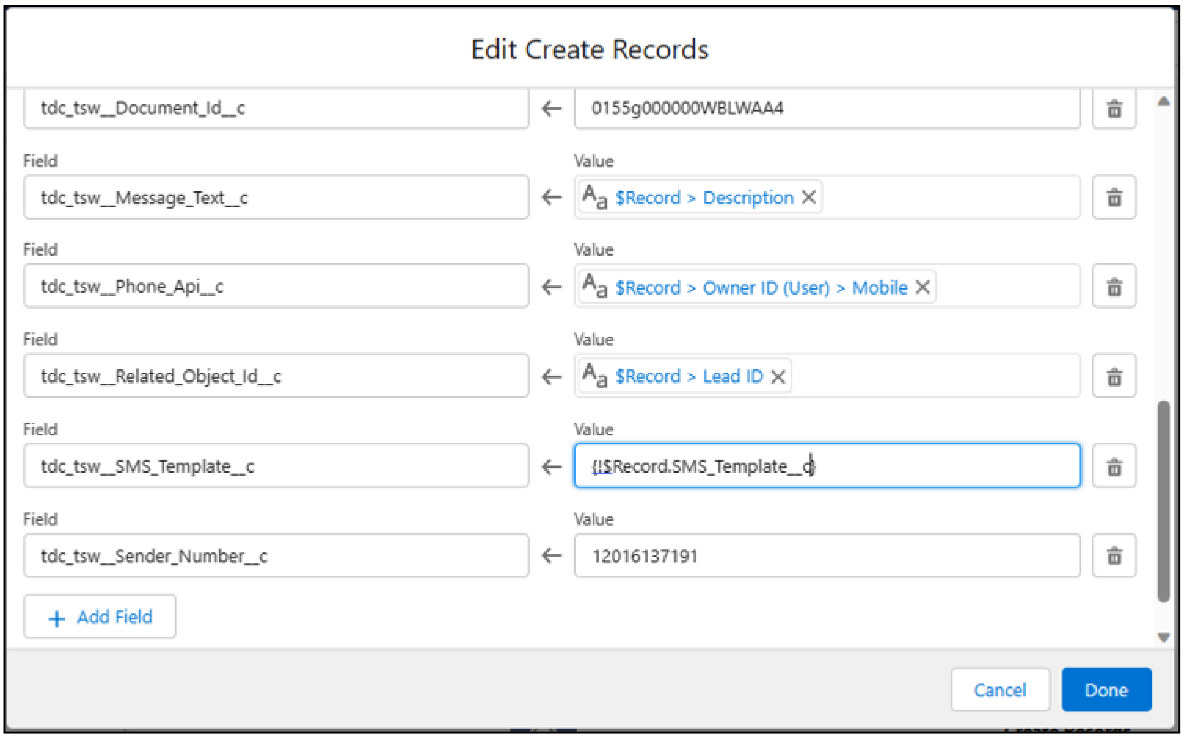
Configure Start

If you create a flow that's triggered when a record is updated, we recommend first defining entry conditions. Then select the **Only when a record is updated to meet the condition requirements** option for When to Run the Flow for Updated Records.

Condition Requirements: All Conditions Are Met (AND)

	Field	Operator	Value
	SMS_Template__c	Is Null	False
AND	SMS_Template__c	Is Changed	True

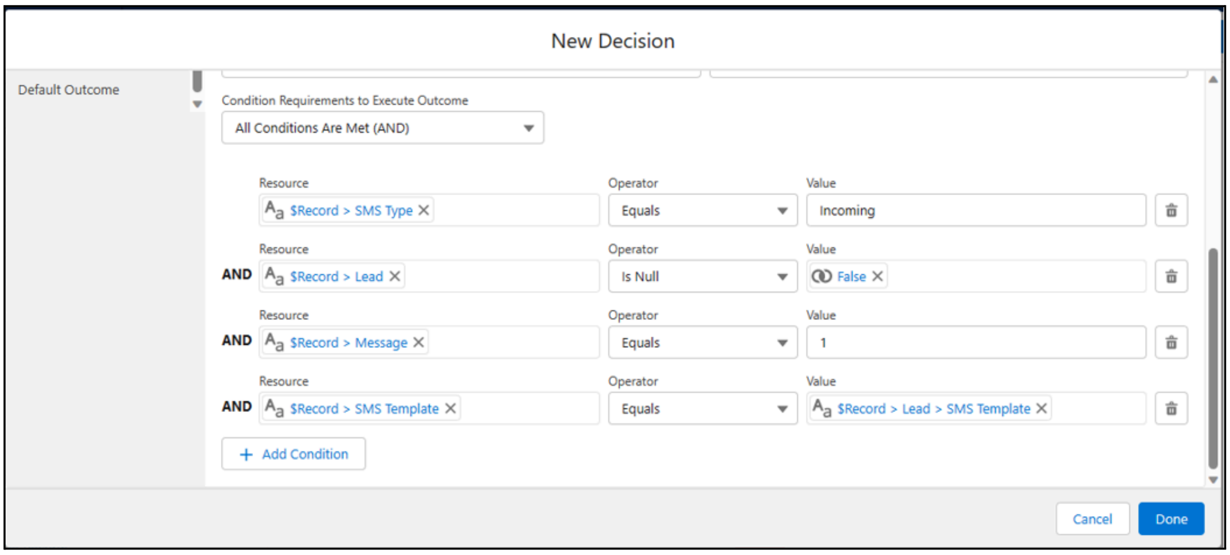
Fig: 25 Master Send SMS Handler



Field	Value
tdc_tsw__Document_Id__c	0155g000000WBLWAA4
tdc_tsw__Message_Text__c	$Record > Description
tdc_tsw__Phone_Api__c	$Record > Owner ID (User) > Mobile
tdc_tsw__Related_Object_Id__c	$Record > Lead ID
tdc_tsw__SMS_Template__c	{!$Record.SMS_Template__c}
tdc_tsw__Sender_Number__c	12016137191

Fig: 25.1 Master Send SMS Handler

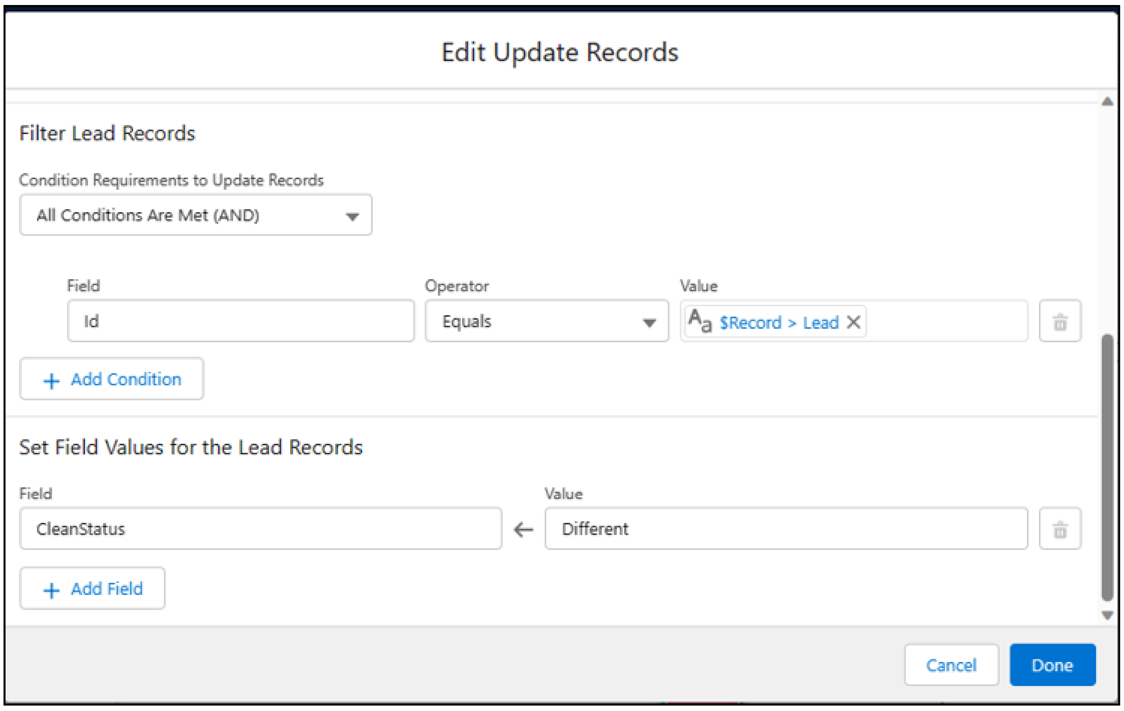
When dealing with an iText that has multiple answers, and each response requires triggering a different template, writing repetitive APEX code for each possible answer becomes challenging. Instead, a more efficient approach is adopted. We simply set the SMS_Template for the contact, and then let our other Flows handle the rest of the process, eliminating the need for redundant code and streamlining the workflow.



Condition Requirements to Execute Outcome: All Conditions Are Met (AND)

	Resource	Operator	Value
	$Record > SMS Type	Equals	Incoming
AND	$Record > Lead	Is Null	False
AND	$Record > Message	Equals	1
AND	$Record > SMS Template	Equals	$Record > Lead > SMS Template

Fig: 25.2 Master Send SMS Handler


Edit Update Records

Filter Lead Records

Condition Requirements to Update Records: All Conditions Are Met (AND)

Field	Operator	Value
Id	Equals	$Record > Lead

Set Field Values for the Lead Records

Field	Value
CleanStatus	Different

Fig: 25.3 Modify a field in the Lead Object within the Master Send SMS Handler.

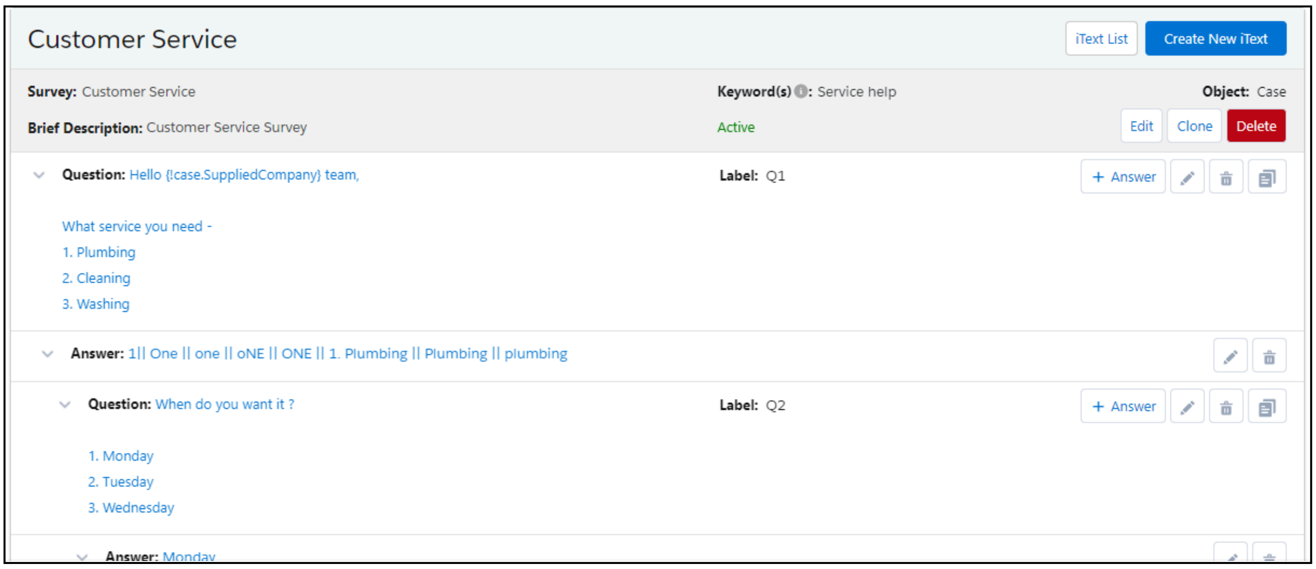
iTexts

The 360SMS iText structure is a robust feature that automates the entire Question/Answer dialog without relying on Flows or Templates. It seamlessly stores answers and their related questions in the SMS History, while also saving them in an iText Response Object for enhanced reporting and automation capabilities. iTexts offer the flexibility for a question to have multiple answers, leading to additional questions with their own set of multiple answers, allowing for as many branched questions and answers as needed.

iTexts can be triggered either through traditional Flows using the methods described in Method #1 or Method #2, or they can be automatically triggered by defining an inbound keyword.

iTexts work exceptionally well when presenting clear answer choices, such as "Reply YES or NO" or using multiple-choice answers where the call-to-action is to "Reply with a number" or "Reply with a letter or combination of letters."

Lastly, iText replies can easily update field values in Salesforce Objects or initiate other actions using Flows on the SMS History or iText Response Object to examine the incoming answer to a specific question. This powerful feature streamlines the question-and-answer process and provides seamless integration with Salesforce Objects and automated actions.


Customer Service

Survey: Customer Service

Brief Description: Customer Service Survey

Keyword(s): Service help

Active

Object: Case

Question: Hello {!case.SuppliedCompany} team,

What service you need -
1. Plumbing
2. Cleaning
3. Washing

Label: Q1

Answer: 1|| One || one || oNE || ONE || 1. Plumbing || Plumbing || plumbing

Question: When do you want it ?

1. Monday
2. Tuesday
3. Wednesday

Label: Q2

Answer: Monday

Fig: 26 iText

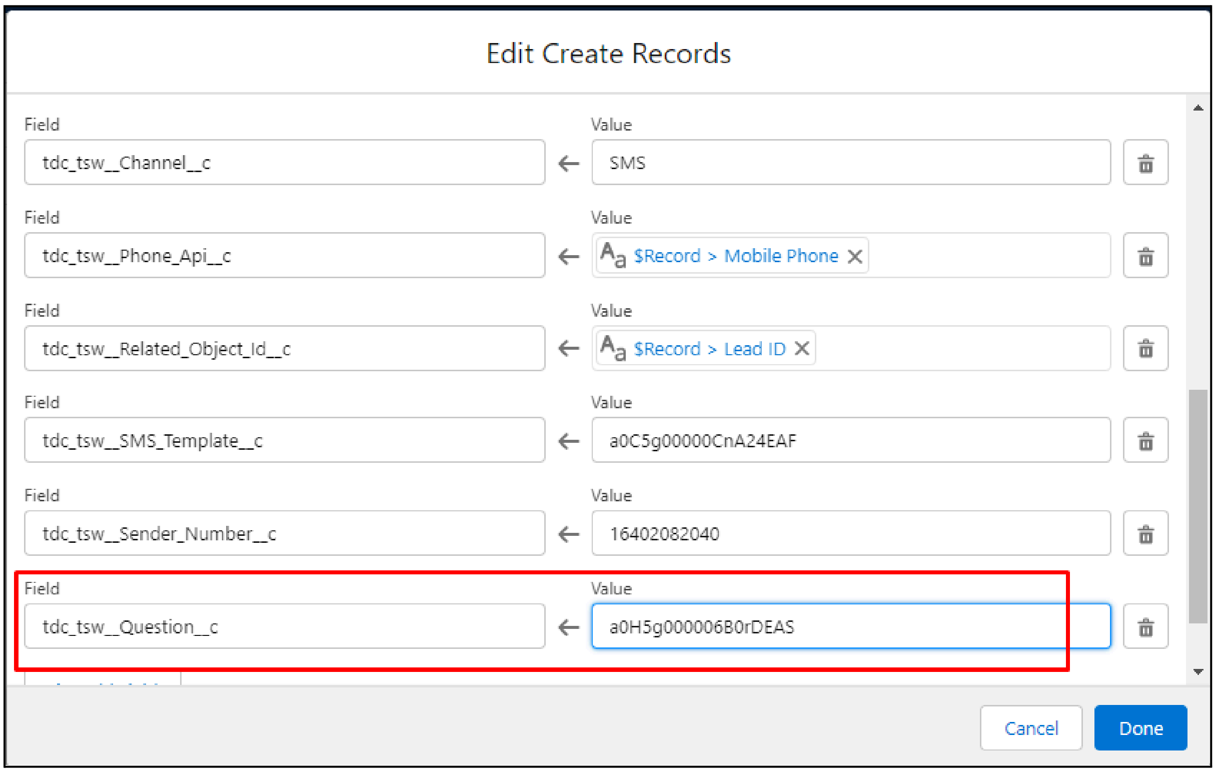
Using Salesforce Flows

The Salesforce Flow technology is worth a brief discussion as it offers significantly more capabilities than Flows. It provides the ability to incorporate complex business logic, utilize variables, look up records, loop through records, delete records, and more.

In the following examples, we demonstrate the process of receiving an email address through an Incoming SMS in response to a template message that prompts the user for their email address to look them up. In this workflow, we encounter a new unknown number entering our Salesforce system. Previously, in Flows, we have created a new Lead record for this Incoming SMS and begun asking questions to complete the record, such as Email, Name, and Company. However, in this case, we utilize a Flow called from a Flows to take the email address and perform a Record Lookup against the Contact Object. If a matching record is found, we send back a template that says, "Found you {!Contact.Name}!" and then proceed with the original keyword that initiated the entire process.

Figure 26 demonstrates how a regular Flows can call a Flow while passing in parameters gathered from the SMS_History record. In Figure 27, we see the details of the Flow, where it:

- Looks up the email from the Contacts Object using the "Get Records" functionality of Flows.
- If a contact is found, it re-links all the SMS_History that was linked to the dummy lead record, which was initially created when the SMS from the unknown number first came in.
- Finally, we can send the outbound SMS reply either through the same APEX methods as a Flows uses, or in this case, we use the technique where we only update the Contact.SMS_Template field, which, in turn, triggers our Master SEND SMS Handler. This approach allows for seamless communication and efficient handling of responses.


Edit Create Records

Field	Value
tdc_tsw__Channel__c	SMS
tdc_tsw__Phone_Api__c	$Record > Mobile Phone
tdc_tsw__Related_Object_Id__c	$Record > Lead ID
tdc_tsw__SMS_Template__c	a0C5g00000CnA24EAF
tdc_tsw__Sender_Number__c	16402082040
tdc_tsw__Question__c	a0H5g000006B0rDEAS

Fig: 27 The iText question ID (pertaining to the first question of the respective iText) is required.