📲 SMS Magic with Salesforce Integration – Complete Guide

🧩 Introduction

In today's fast-paced business world, staying connected with customers is critical. **SMS Magic** makes this possible by enabling businesses to send automated, personalized text messages right from Salesforce. Whether it's appointment reminders, lead follow-ups, support updates, or promotional campaigns — integrating SMS Magic with Salesforce ensures timely and efficient communication.

This guide walks you through how to seamlessly integrate SMS Magic with Salesforce, covering setup, automation, and best practices.

🔍 What is SMS Magic?

SMS-Magic Converse is a messaging software solution. It is simple to deploy, easy to use, and delivers the most powerful messaging capabilities available today.

SMS-Magic Converse empowers users of text messaging to engage, win, and retain more buyers, creating strong relationships that drive sustainable competitive advantage.

Key Features:

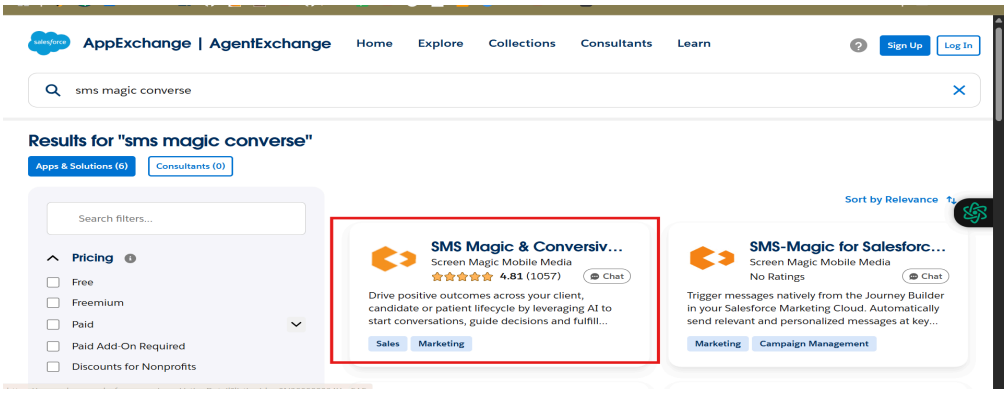
- One-on-one & bulk messaging
- SMS automation with Flows & Process Builder
- Multi-channel communication (SMS, WhatsApp, Facebook)
- Messaging history tracking
- Template management with merge fields

🛠️ How to Integrate SMS Magic with Salesforce?

🔧 Step-by-Step Integration Guide

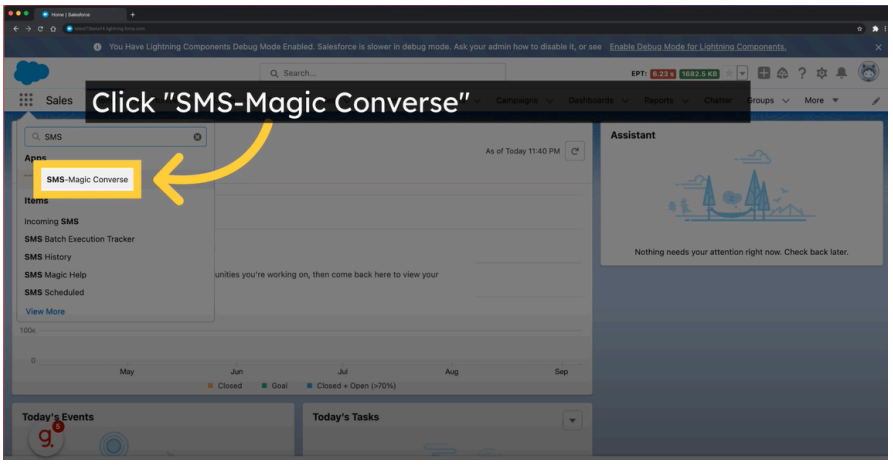
Step 1: Install SMS Magic from AppExchange

1. Log in to your **Salesforce Org**
2. Go to [AppExchange]
3. Search for **"SMS Magic Converse"**
4. Click **"Get It Now"**
5. Choose your environment **(Production/Sandbox)**
6. Install for Admins Only or All Users as needed.
7. Ensure the appropriate users have access to SMS Magic objects and features. You can use the permission sets provided by SMS Magic or configure manually as needed.



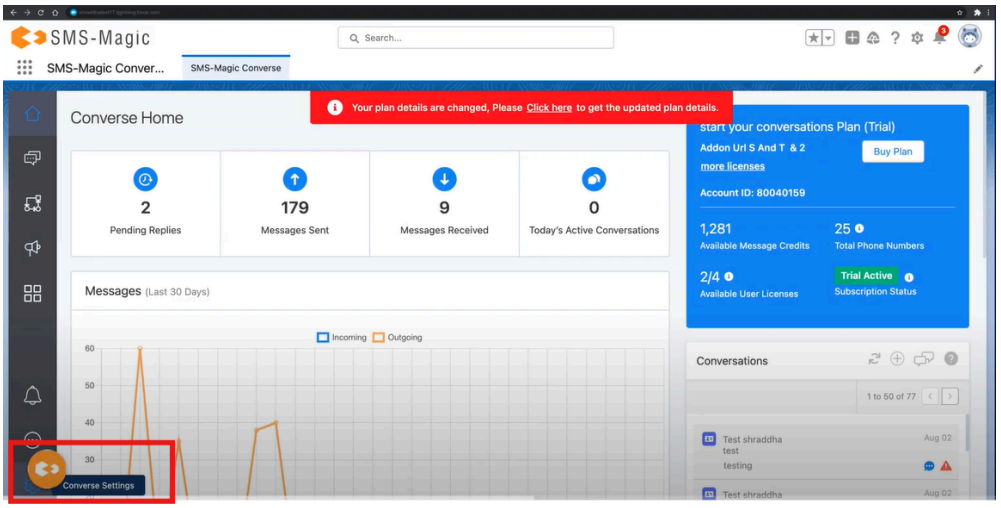
Step 2: Configure SMS Magic in Salesforce

1. Open the **SMS Magic Converse App** from App Launcher



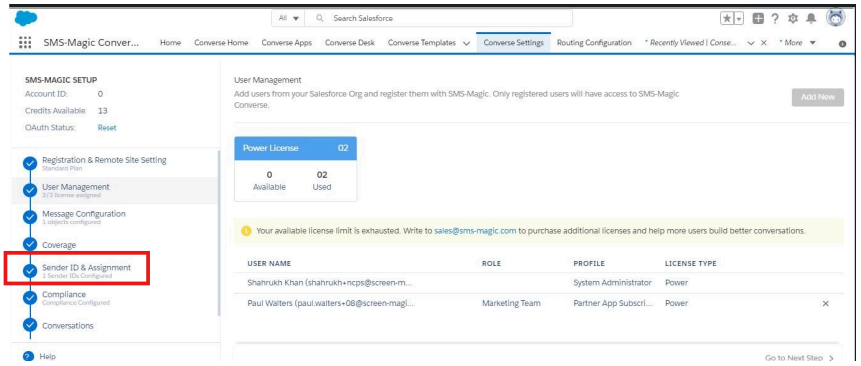
2. Now **Converse home page** will appear
3. Go to **Converse Settings**
    - Add your **SMS Gateway (e.g., Twilio)**
    - Map **SMS objects** to Salesforce records

- Enable Message Templates



Step 3: Configure Sender ID (SMS Channel)

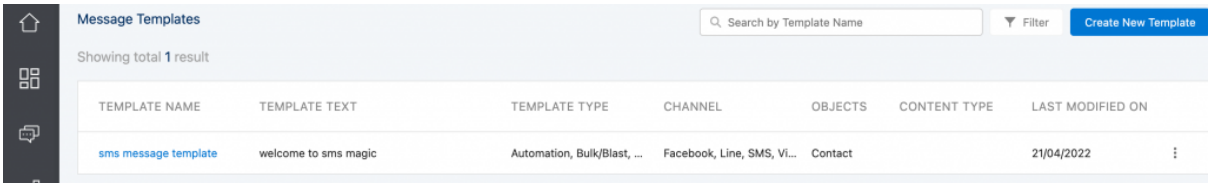
1. In the SMS Magic app, go to **Configure Channels**.
2. Add your phone number (Sender ID).
3. Complete verification if required (OTP/approval).
4. Set this channel as the default.



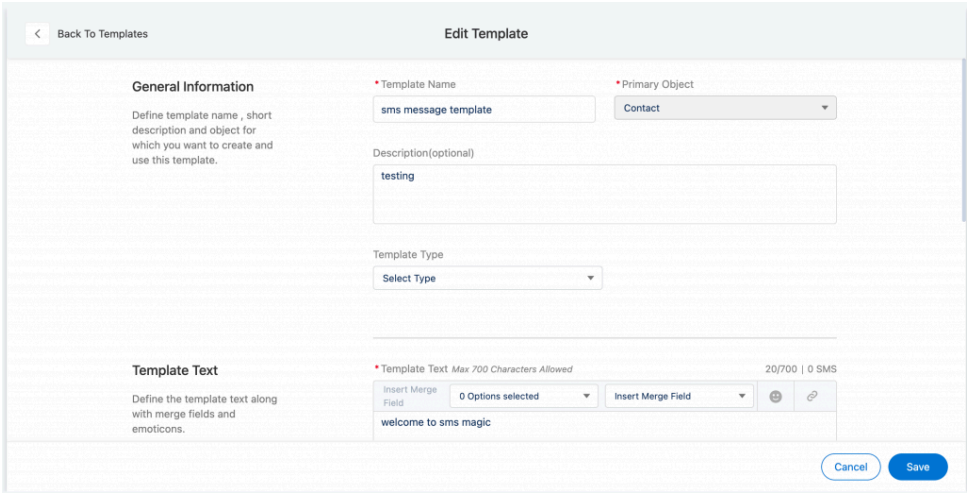
Step 4: Create SMS Templates

You can create new templates, edit existing templates, change a template owner, and attach a template to an object. Also, it is easy to add merge fields to the templates in order to personalize each of the outgoing messages.

1. In the **SMS-Magic Converse** application.
2. Click on the **Converse Templates** tab. The Converse Templates page appears:



3. Click on the '**Create New Template'** button. The New Template pop-up window appears. You can enter the basic information.

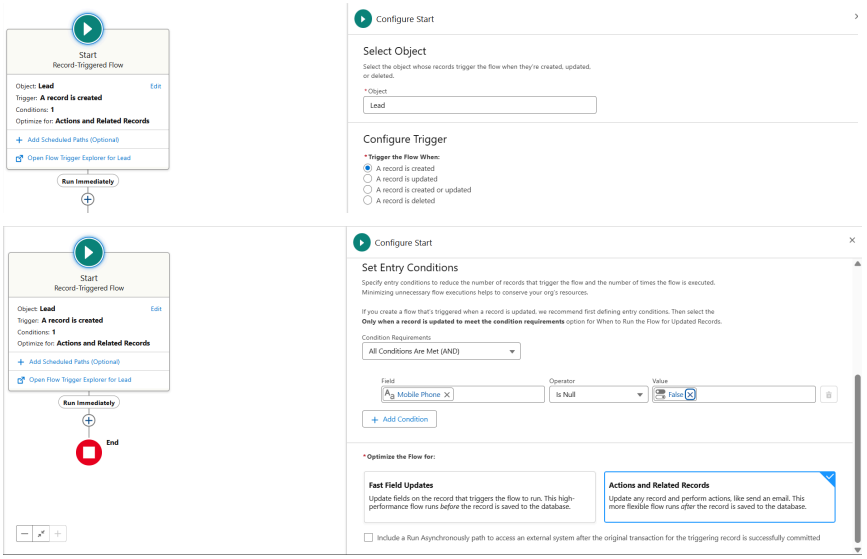


4. In the Basic Information section, enter the template name in the '**Template Name**' field and select the **primary object** from the Primary Object drop-down list.
5. Type the **description** (Type a small summary to describe the template) in the description field.
6. Select the template type.
7. Enter the template text (Type the default template text. Select the fields that you want to use as merge fields In **Insert Merge Field** within the template text.)
8. Click **Create** once all details are filled.

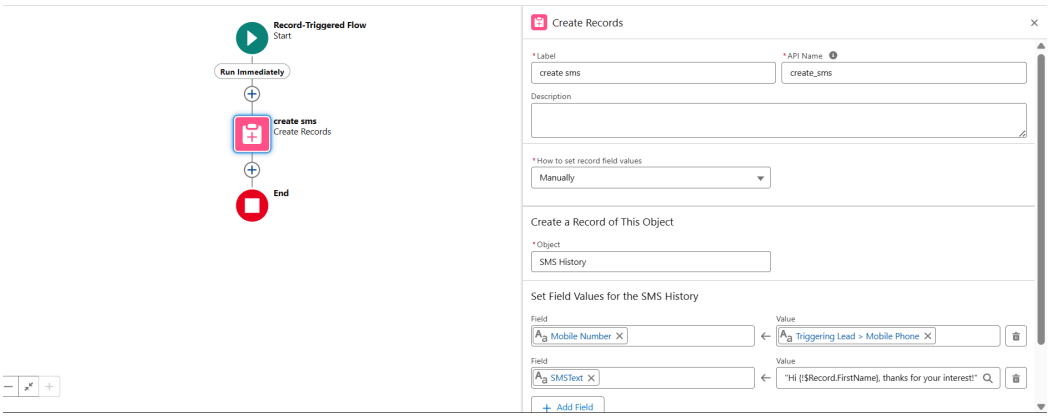
Step 5: Automate SMS with Flows

💬 **Example: Send Welcome SMS on Lead Creation**

- Create a Record-Triggered Flow
- Object: Lead
- Trigger: When a record is created



- Entry Condition: Mobile Phone IS NOT NULL
- Create SMS Record
- Element Type: Create Records
- Object: **smagicinteract__smsMagic__c**
- Set these fields:
    - **smagicinteract__PhoneNumber__c**– Lead.Mobile Phone
    - **smagicinteract__SMSText__c**– "Hi {!Lead.FirstName}, thanks for your interest!"
    - **smagicinteract__SenderId__c**- "YourSenderID123" *(replace with your actual sender ID)*
    - **smagicinteract__disableSMSOnTrigger__c** - 0



- Add the required criteria in the flow and click on the **'Save'** button
- Click on **Run** to test the flow
- Click on **Activate** to activate the message flow

🚀Using SMS Magic in Apex (Salesforce Development)

If you want to send SMS messages programmatically within Salesforce using Apex, **SMS Magic** provides an API-based approach to send, receive, and automate messages.

Here's how you can use Apex code to integrate SMS Magic.

📌Steps to Send SMS using SMS Magic in Apex

✅Step 1: Create an SMS Record in Salesforce

It uses a custom object (**smagicinteract__smsMagic__c**) and its fields to store SMS details for subsequent processing, potentially by a trigger or external service.

```
public static void createSmsRecord(String phoneNumber, String message, String senderId) {
    List<smagicinteract__smsMagic__c> smsObjectList = new List<smagicinteract__smsMagic__c>();
    smagicinteract__smsMagic__c smsObject = new smagicinteract__smsMagic__c();

    smsObject.smagicinteract__SenderId__c = senderId; // Set your desired sender ID
    smsObject.smagicinteract__PhoneNumber__c = phoneNumber;
    smsObject.smagicinteract__SMSText__c = message; // The text of the SMS message
    smsObject.smagicinteract__disableSMSOnTrigger__c = 0; // Allow trigger to send SMS
    smsObject.smagicinteract__external_field__c = smagicinteract.ApexAPI.generateUniqueKey();

    smsObjectList.add(smsObject);
    Database.insert(smsObjectList,false);
}
```

✅Step 2: Send Bulk SMS using Apex

Sometimes you need to send SMS to multiple records at once — like all Leads from a Campaign, or all Contacts with open Opportunities. You can easily do this using bulkified Apex code with SMS Magic.

NOTE- 📜 **SMS History Creation**

In our solution, the **createSMSHistory** function is responsible for saving the details of each SMS message sent. It inserts records of the **smagicinteract__smsMagic__c** custom object, which holds the SMS details, into the database. The details include the contact's mobile number, the SMS message, the sender's phone number, and other relevant information.

```
public class SendSMS {
   public static void sendGenericSMSToContacts(Set<String> contactIds) {
      // Fetch all required Contacts
      Map<Id, Contact> contactMap = new Map<Id, Contact>([
         SELECT Id, OwnerId, MobilePhone, Account.Owner.IsActive, Do_not_sms__c, SMS_Opt_Out__c
         FROM Contact
         WHERE Id IN :contactIds
            AND Account.Owner.IsActive = TRUE
            AND Do_not_sms__c = FALSE
            AND SMS_Opt_Out__c = FALSE
      ]);

      // Collect all unique OwnerIds
      Set<Id> ownerIds = new Set<Id>();
      for (Contact con : contactMap.values()) {
         if (con.OwnerId != null) {
            ownerIds.add(con.OwnerId);
         }
      }
      Map<Id, String> ownerPhoneMap = new Map<Id, String>();
      for (User userObj : [SELECT Id, Phone FROM User WHERE Id IN :ownerIds AND Phone != NULL]) {
         ownerPhoneMap.put(userObj.Id, userObj.Phone);
      }
     // Prepare SMS records
      List<smagicinteract__smsMagic__c> smsList = new List<smagicinteract__smsMagic__c >();

      for (Contact contact : contactMap.values()) {
         if (contact.MobilePhone != NULL) {
            smagicinteract__smsMagic__c smsRecord = new smagicinteract__smsMagic__c();
            smsRecord.smagicinteract__Contact__c= contact.Id;
            smsRecord.OwnerId = contact.OwnerId;
            smsRecord.smagicinteract__PhoneNumber__c = contact.MobilePhone;
            smsRecord.smagicinteract__Source__c = 'GenericSource';
```

```
            String messageBody = 'We value your input! Your feedback is important to us.';
            messageBody += ' Please take a moment to complete a quick survey to help us improve.';
            messageBody += ' [Survey Link]';
            smsRecord.smagicinteract__SMSText__c = messageBody;

            if (ownerPhoneMap.containsKey(contact.OwnerId)) {
                smsRecord.smagicinteract__SenderId__c = ownerPhoneMap.get(contact.OwnerId);
            }

            smsList.add(smsRecord);
        }
    }
   // Insert records if any
    if (!smsList.isEmpty()) {
        createSMSHistory(smsList);
    }
  }
}
```

✅**This will send SMS to multiple contacts in a single execution.**

✅ **Best Practices for Bulk SMS in Apex**

- **Bulkify logic** - Prevents hitting governor limits
- **Check for empty phone numbers -** Avoids invalid records
- **Set Status = 'Pending' -** Triggers the actual SMS sending
- **Use Test Data in Unit Tests -** Keeps tests reliable and isolated

✅Step 3: Automate SMS using Apex Trigger

This trigger will run when a **Contact's Status field** changes to **"Active"**, an automated **Welcome SMS** should be sent **only once**, not every time the record is updated.

```
trigger ContactStatusTrigger on Contact (after update) {
    Set<Id> contactIdsToSendSMS = new Set<Id>();
for (Contact newCon : Trigger.new) {
        Contact oldCon = Trigger.oldMap.get(newCon.Id);
        if (newCon.Status__c == 'Active' && oldCon.Status__c != 'Active') {
            contactIdsToSendSMS.add(newCon.Id);
        }
    }

    if (!contactIdsToSendSMS.isEmpty()) {
        SMSHandler.sendWelcomeSMS(contactIdsToSendSMS);
    }
}
```

🧠 Helper Class: SMSHandler

```
public class SMSHandler {
    public static void sendWelcomeSMS(Set<Id> contactIds) {
        List<Contact> contacts = [ SELECT Id, FirstName, LastName, MobilePhone, OwnerId  FROM Contact WHERE Id IN :contactIds   AND MobilePhone != NULL ];

        Map<Id, String> ownerPhoneMap = new Map<Id, String>();
        Set<Id> ownerIds = new Set<Id>();
        for (Contact con : contacts) {
            if (con.OwnerId != null) {
                ownerIds.add(con.OwnerId);
            }
        }

        for (User usr : [SELECT Id, Phone FROM User WHERE Id IN :ownerIds AND Phone != NULL]) {
            ownerPhoneMap.put(usr.Id, usr.Phone);
```

```
    }
List<smagicinteract__smsMagic__c> smsList = new List<smagicinteract__smsMagic__c>();
    for (Contact con : contacts) {
        smagicinteract__smsMagic__c sms = new smagicinteract__smsMagic__c();
        sms.smagicinteract__Contact__c = con.Id;
        sms.smagicinteract__PhoneNumber__c = con.MobilePhone;
        sms.smagicinteract__SenderId__c = ownerPhoneMap.get(con.OwnerId);
        sms.smagicinteract__SMSText__c = '🎉 Welcome aboard ' + con.FirstName + '! Your profile is now active. Let’s achieve great things together.';
        sms.smagicinteract__Source__c = 'ContactActivation';
        sms.smagicinteract__disableSMSOnTrigger__c = 0;
        sms.smagicinteract__external_field__c = smagicinteract.ApexAPI.generateUniqueKey();
        smsList.add(sms);
    }

    if (!smsList.isEmpty()) {
        insert smsList;
    }
  }
}
```

✅Step 4: Check SMS Status using SOQL

Once you’ve sent SMS messages using Apex or Flows, you’ll want to **track their delivery status**. SMS Magic stores the status in the **smsmagic__SMS_History__c** object, which you can query using **SOQL** to monitor message delivery, failure, or processing state.

```
List< smagicinteract__smsMagic__c> smsLogs = [
  SELECT Id, smagicinteract__PhoneNumber__c, smagicinteract__SMSText__c, CreatedDate
  FROM smagicinteract__smsMagic__c
  WHERE CreatedDate = LAST_N_DAYS:7
  ORDER BY CreatedDate DESC
];}
```

✅This allows developers to track the delivery status of messages in Salesforce.