Schedule SMS Messages

With the Salesforce.com scheduler, you can schedule a SMS job for a particular day and time. We can accomplish this by employing the **Schedulable** Apex interface.

The **Schedulable** interface contains one method that must be implemented, the method is **Execute**.

```
global void execute(SchedulableContext sc){}
```

In this method you can write your business logic, and then follow the steps given below and schedule the Apex class.

Step 1 – Goto to Setup –> App Setup –> Develop –> Apex classes.

Step 2 – Click the Schedule Apex button, complete the form, and save

In the figure above, we can see that the class is ready to execute. This class is set to run each day and send SMS messages according to the business logic.

Step 3 – Choose the type of scheduling

Prepare SMS at time of schedule

To schedule an SMS message to be sent to a group of numbers, first create the instance of **smagicinteract__Scheduled_SMS__c** object and attach the required fields such as **smagicinteract__MobilePhone__c**, **smagicinteract__status__c**, **smagicinteract__Scheduled_Date__c**, **smagicinteract__SMSText__c**. Any records that are scheduled will be picked at the time of distribution. Note that smagicinteract above is a package prefix which will differ from package to package.

The advantage of this type is that you can cancel the schedule by simply deleting the records. The following sample code demonstrates this more clearly.

```
List conList = [select Id,FirstName, LastName, MobilePhone, Name from
Contact];
Date scheduleDate = Date.valueOf('2011-08-10');
List scheduleSMSList = new
List();
if(conList != null){
for(Contact contact : conList){
```

```
smagicinteract__Scheduled_SMS__c scheduleSMSObject = new
smagicinteract__Scheduled_SMS__c();
if(contact.MobilePhone != null){
scheduleSMSObject.smagicinteract__MobilePhone__c = contact.MobilePhone;
scheduleSMSObject.smagicinteract__jobId__c = '1';
scheduleSMSObject.smagicinteract__status__c = 'Schedule';
scheduleSMSObject.smagicinteract__Scheduled_Date__c = scheduleDate;
scheduleSMSObject.smagicinteract__SMSText__c = 'Test Of Schedule';
}
scheduleSMSList.add(scheduleSMSObject);
}
insert scheduleSMSList;
}
```

After this, use a **Schedulable** class to pickup the **scheduled_SMS__c** object record—according to **scheduleDate** and send the SMS messages by creating the **SMS_Magic__c** object—or us **SMSPushAPI**.

Message rendered at schedule execution

With this approach, you simply schedule on a particular object. We do not create the object of **smagicinteract__Scheduled_SMS__c**, but only schedule on a particular object. You can write one Schedulable class that will pick up particular object records according to the timestamps of each. Render SMS drafts using the template created for that object and the SMS messages will be sent. To cancel a scheduled SMS distribution, you'll need to delete the scheduled job itself. The following sample code will schedule SMS messages from a custom object named **Time sheet**.

```
global class TodaysScheduleSMS implements Schedulable {
String day = '';
String query = '';
List todaysScheduleList = null;
String smsText = '';
String status = '';
String week = '';
global void execute(SchedulableContext sc){
// Here we can get today's day.
day = DateTime.now().format('EEEEE');
todaysScheduleList = [select SMS_Template__c, Status__c, Day__c, Week__c from
Time_Sheet_Schedule__c where Day__c =:day ];
if(todaysScheduleList.size() > 0){
for(Time_Sheet_Schedule__c timeSheetScheduleObj : todaysScheduleList){
status = timeSheetScheduleObj.Status__c;
week = timeSheetScheduleObj.Week__c;
List smsTemplateList = [select
smagicinteract__Text__c from smagicinteract__SMS_Template__c where Id
=:timeSheetScheduleObj.SMS_Template__c];
if(smsTemplateList != null){
smsText = smsTemplateList[0].smagicinteract__Text__c;
}
```

```
}
/* write here code to send SMS to number */
}
}
}
```

There are limits on how many callouts and DML statements you can execute from a scheduled job. Only 10-20 SMS messages can be sent in one job. There is a workaround to exceed this limit, in which you can use a batchable class – as outlined below.

```
global class SendScheduleSMSBatch implements Database.Batchable,
Database.AllowsCallouts {

global SendScheduleSMSBatch(){
}
global Database.QueryLocator start(Database.BatchableContext BC){
String day = DateTime.now().format('EEEEE');
String scheduleStatus = 'Schedule';
String query = 'select SMS_Template__c, Status__c, Day__c, Week__c from
Time_Sheet_Schedule__c where Day__c =:day';
return Database.getQueryLocator(query);
}

global void execute(Database.BatchableContext BC, List scope){
if(scope.size() > 0){
for(Time_Sheet_Schedule__c timeSheetScheduleObj : todaysScheduleList){
status = timeSheetScheduleObj.Status__c;
week = timeSheetScheduleObj.Week__c;
List smsTemplateList = [select
smagicinteract__Text__c from smagicinteract__SMS_Template__c where Id
=:timeSheetScheduleObj.SMS_Template__c];
if(smsTemplateList != null){
smsText = smsTemplateList[0].smagicinteract__Text__c;
}
}
/* write here code to send SMS to number */
}
global void finish(Database.BatchableContext BC){
// send batch execution email;
}
}
```

From the Schedulable class, you need to call a batch class:

```
SendScheduleSMSBatch batchSMS = new SendScheduleSMSBatch();
Database.executeBatch(batchSMS);
```

An example test class for this batch class is given below.

```
@isTest
private class SendScheduleSMSBatchTest {
```

```
static Time_Sheet_Schedule__c timeSheetSchedule = null;
static smagicinteract__SMS_Template__c smsTpl= null;
static void setupTest(){

smsTpl = new smagicinteract__SMS_Template__c();
smsTpl.smagicinteract__Text__c = 'Test of Schedule SMS';
smsTpl.smagicinteract__Name__c = 'Schedule SMS Template'
smsTpl.smagicinteract__ObjectName__c = 'Time_Sheet_Schedule__c';
insert smsTpl;
timeSheetSchedule = new Time_Sheet_Schedule__c();
timeSheetSchedule.SMS_Template__c= smsTpl.Id;
timeSheetSchedule.Status__c = 'Scheduled';
timeSheetSchedule.Day__c = 'Monday';
timeSheetSchedule.Week__c = 'this';
insert timeSheetSchedule;
}
static void testMethod test_ScheduleSMSBatch(){
Test.StartTest();
setupTest();
SendScheduleSMSBatch sscb = new SendScheduleSMSBatch();
ID batchprocessid = Database.executeBatch(sscb);
Test.stopTest();
List timeSheetScheduleList = [select Id, Status__c
from Time_Sheet_Schedule__c where Id =:timeSheetSchedule.Id];
system.assertEquals(timeSheetScheduleList[0].Status__c, 'Sent');
}
}
```