SMS Technical Implementation Document

Ping4SMS Integration with DLT Registration

Document Control

- **Version:** 1.0
- **Date:** March 29, 2025
- **Status:** Implementation Guide

1. Introduction

This technical document outlines the implementation of SMS messaging services using Ping4SMS with DLT (Distributed Ledger Technology) registration. The document provides detailed specifications, configuration parameters, and implementation guidelines for developers to integrate SMS capabilities into applications.

2. DLT Registration Process

2.1 Overview

DLT registration is mandatory for sending commercial SMS messages in India. It ensures regulatory compliance with TRAI guidelines.

2.2 Registration Steps

1. Register as a Principal Entity (PE) on the DLT platform
2. Select a telecom operator and complete registration
3. Obtain PE ID upon successful registration
4. Register your SMS content templates
5. Register your sender IDs
6. Wait for approval (typically takes 24-48 hours)

2.3 Template Registration

Templates must be registered with the following details:

- Template content with variables marked as {#var#}
- Communication category (Promotional/Transactional/Service)
- Use case description

3. Ping4SMS Configuration

3.1 Account Details

- **URL:** www.ping4sms.com
- **Username:** xxxxxx
- **Password:** xxxxxx
- **API Key:** xxxxxxxxxxxxx

3.2 API Endpoints

- **Message Sending API:** GET http://site.ping4sms.com/api/smsapi?key={apikey}&route={route}&sender=PINGSM&number={Number}&sms={Message}&templateid={templateid}
- **Delivery Report API:**GET http://site.ping4sms.com/api/dlrapi?key={apikey}&messageid={UniqueID}
- **Credit Check API:** GET https://site.ping4sms.com/api/creditapi?key={apikey}&route={Route}

4. Implementation Architecture

4.1 System Components

- **Database Handler:** Manages data operations for contacts and campaigns
- **SMS Controller:** Handles SMS sending logic and API interactions
- **Configuration Service:** Manages API credentials and settings
- **Dialler Service:** Orchestrates the SMS sending process

4.2 Data Flow

1. Load campaign contacts from database
2. Validate phone numbers and remove duplicates

3. Process message templates with dynamic content
4. Send SMS through Ping4SMS API
5. Retrieve delivery reports
6. Update campaign status in database

5. Code Implementation

5.1 SMS Sending Function

Request Parameters:

- sender (string): The sender ID obtained from DLT.
- number (string): Recipient's phone number.
- message (string): Message content fetched from DLT.
- templateid (string): DLT-approved template ID.
- baseUrl (string): API base URL.
- apiKey (string): API authentication key.
- route (string): SMS route.
- placeholders (Dictionary<string, string>): Key-value pairs to replace placeholders in the message.

Response:

- Returns a string containing the message ID if successful, otherwise null.

Code :

```
private static string SendSMS(string sender, string number, string message,
   string templateid, string baseUrl, string apiKey, string route,
   Dictionary<string, string> placeholders)
{
   try
   {
      // Replace placeholders in message
      foreach (var placeholder in placeholders)
      {
         message = message.Replace(placeholder.Key, placeholder.Value);
      }

      // Construct API URL
```

```
        string url = $"{baseUrl}?key={apiKey}&route={route}" +

$"&sender={sender}&number={number}&sms={HttpUtility.UrlEncode(message)}&templat
eid={templateid}";

        // Send HTTP request
        HttpWebRequest myReq = (HttpWebRequest)WebRequest.Create(url);
        myReq.Method = "GET";

        using (HttpWebResponse myResp = (HttpWebResponse)myReq.GetResponse())
        using (StreamReader respStreamReader = new
StreamReader(myResp.GetResponseStream()))
        {
            string responseString = respStreamReader.ReadToEnd();
            Console.WriteLine($"SMS API Response for {number}: {responseString}");

            if (long.TryParse(responseString, out _))
            {
                return responseString; // It's a valid numeric message ID
            }
            Console.WriteLine($"Unexpected API response format: {responseString}");
            return null;
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error sending SMS to {number}: {ex.Message}");
        return null;
    }
}
```

5.2 Delivery Report Function

Request Parameters:

- messageId (string): Unique message ID received from the SMS API.
- smsSettings (IOptions): Contains API key for authentication.

Response:

- Returns a string with the delivery report details.

```
private static string GetSMSStatus(string messageId, IOptions<SmsSettings> smsSettings)
{
    try
    {
        var settings = smsSettings.Value;

        string url =
$"https://site.ping4sms.com/api/dlrapi?key={settings.ApiKey}&messageid={messageId}";

        HttpWebRequest myReq = (HttpWebRequest)WebRequest.Create(url);
        myReq.Method = "GET";

        using (HttpWebResponse myResp = (HttpWebResponse)myReq.GetResponse())
        using (StreamReader respStreamReader = new
StreamReader(myResp.GetResponseStream()))
        {
            return respStreamReader.ReadToEnd();
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error fetching SMS status for Message ID {messageId}:
{ex.Message}");
        return "Error fetching status";
    }
}
```

5.3 Database Update Function

Request Parameters:

- campaignid (string): ID of the campaign.
- contact_id (string): ID of the contact.
- phoneno (string): Contact phone number.

Response:

- Returns true if the database update was successful, otherwise false.

```
private static bool UpdateDatabase(IDbHandler dbHandler, string campaignid,
    string contact_id, string phoneno)
{
    try
    {
        var result = dbHandler.ExecuteScalar(
            "EXEC UpdateCampaignContactAndPaymentDetails @campaign_id, @contact_id,
@phoneno",
            new Dictionary<string, object>
            {
                { "@campaign_id", campaignid },
                { "@contact_id", contact_id },
                { "@phoneno", phoneno }
            });

        return result != null;
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Database Update Failed for {phoneno}: {ex.Message}");
        return false;
    }
}
```

6. Configuration Settings

6.1 appsettings.json Configuration

```
{
 "SmsSettings": {
  "BaseUrl": "http://site.ping4sms.com/api/smsapi",
  "ApiKey": "xxxxxxxxxxxxxxx",
  "Route": "2"
 },
 "Database": {
  "ConnectionString": "Your_Database_Connection_String"
 }
}
```

6.2 Loading Configuration in Code

```
var builder = new ConfigurationBuilder();
builder.SetBasePath(Directory.GetCurrentDirectory())
      .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true);
IConfiguration config = builder.Build();
```

7. SMS Processing Flow

7.1 Main Process Function

The core SMS processing function SMS_contacts_dialAsync handles:

- Loading campaign contacts from database
- Determining workspace context
- Processing each contact
- Avoiding duplicate messages
- Template processing with placeholders
- Sending SMS messages
- Retrieving delivery reports
- Updating campaign status

7.2 Error Handling

- Duplicate number detection
- Empty phone number validation
- API response validation
- Exception handling at multiple levels
- Logging of errors and responses

8. SMS Message Format

8.1 Template Structure

Templates must follow the approved DLT format:

Dear Customers, Welcome to PING4SMS. {#var#}

8.2 Placeholder Replacement

```
var placeholders = new Dictionary<string, string>
{
   { "{#var#}", "click2go.ai/#Features" }
};
```

9. Integration Testing

9.1 Testing Checklist

- Verify DLT registration is complete
- Confirm template approval and template ID
- Test API connectivity
- Validate placeholder replacement
- Check delivery report functionality
- Verify database updates

9.2 Test Cases

1. Single SMS sending
2. Batch SMS sending

3. Error handling for invalid numbers
4. Proper placeholder replacement
5. Delivery report retrieval

10. Troubleshooting

10.1 Common Issues

- **Template Rejection:** Ensure templates comply with DLT guidelines
- **API Connectivity Issues:** Check network and firewall settings
- **Invalid Response Format:** Verify API key and parameters
- **Delivery Failures:** Review delivery reports for carrier-specific issues

10.2 Logging and Monitoring

Implement comprehensive logging:

```
Console.WriteLine($"SMS API Response for {number}: {responseString}");
Console.WriteLine($"Error sending SMS to {number}: {ex.Message}");
```

11. Security Considerations

- Store API credentials securely
- Use HTTPS for all API communications
- Implement rate limiting to prevent abuse
- Follow password security best practices
- Encrypt sensitive data in transit and at rest

12. Production Deployment

12.1 Deployment Checklist

- Complete DLT registration and template approval
- Set up production API credentials
- Configure production database connections
- Implement monitoring and alerting
- Test in staging environment before production release

12.2 Scaling Considerations

- Implement batch processing for large campaigns
- Consider queue-based architecture for high volume
- Optimize database queries and connection pooling
- Set up appropriate timeout configurations