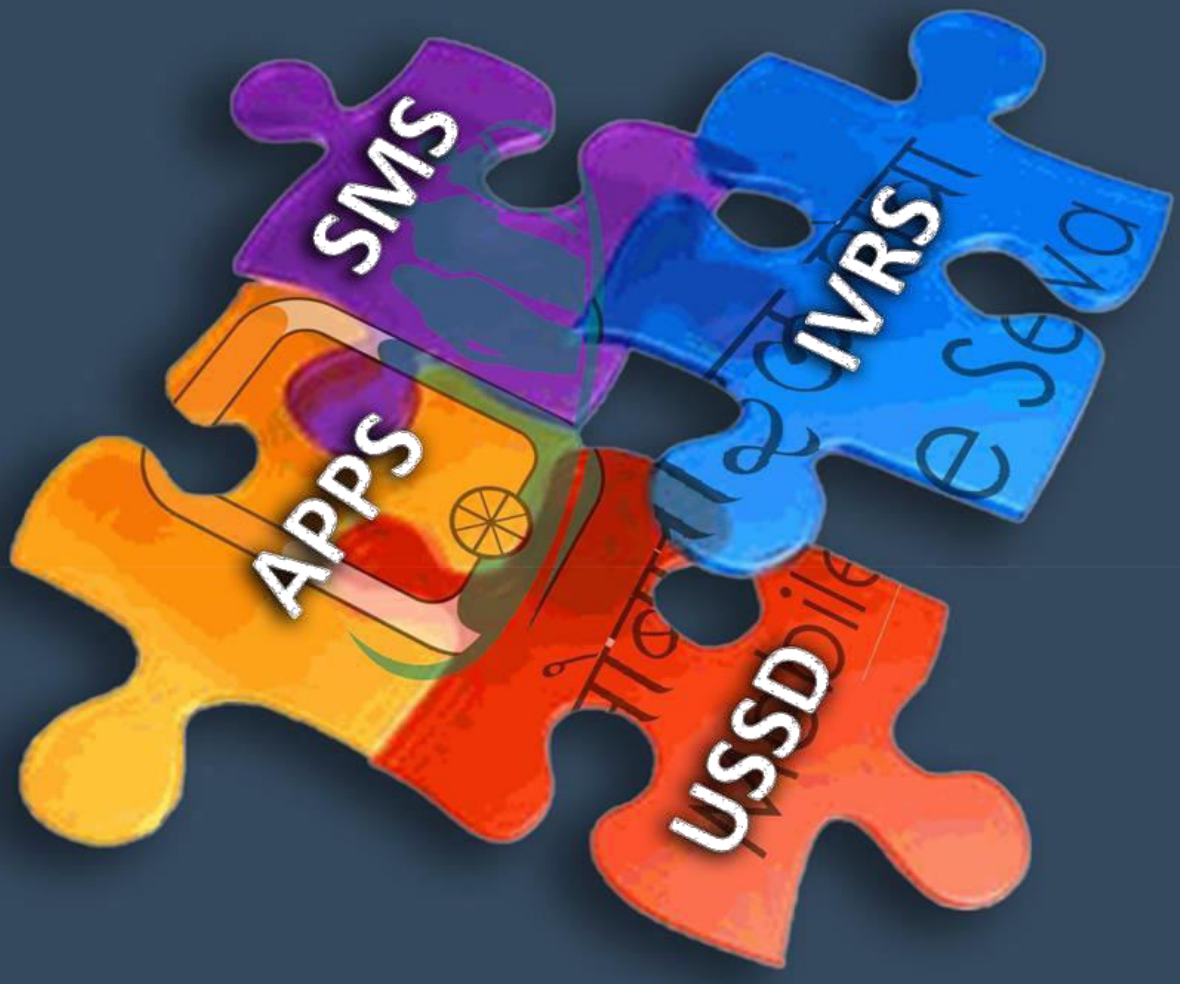
Mobile Seva

MSDP Services Integration Document



CHAPTER 1 : PUSH SMS INTEGRATION 3

OVERVIEW 3
TERMS AND DEFINITIONS 3
USING THE HTTP URL FOR SENDING MESSAGES: 3
PUSH ACCOUNT CREATION 3
PUSH PARAMETER DEFINITIONS: 4
API RESPONSE CODES 4
JAVA EXAMPLE FOR SECURE PUSH 5
.NET (C#) EXAMPLE FOR SECURE PUSH 19
PHP EXAMPLE FOR SECURE PUSH 26

CHAPTER 2 : PULL SMS INTEGRATION 33

OVERVIEW 33
SMS PULL ACCOUNT CREATION 33
SMS PULL PARAMETER DEFINITIONS: 34
EXAMPLE OF PULL REQUEST: 35
FOR SECURE PULL SERVICE: 35
EXAMPLE OF SECURE PULL REQUEST: 35
HOW TO CHOOSE KEYWORDS AND SUB-KEYWORDS? 35

CHAPTER 3 : IVRS PULL INTEGRATION 37

OVERVIEW 37
IVRS PARAMETER DEFINITIONS: 37
EXAMPLE OF IVRS REQUEST: 38

CHAPTER 4 : IVRS PUSH INTEGRATION 39

OVERVIEW 39
IVRS ACCOUNT CREATION 39
TERM AND DEFINATION 40
JAVA EXAMPLE: 40
API RESPONSE CODES 44

CHAPTER 5 : UNSTRUCTURED SUPPLEMENTARY SERVICES DATA (USSD) 45

USSD REQUIREMENT FROM DEPARTMENT 45
JAVA EXAMPLE FOR USSD 46

Chapter 1 : Push SMS Integration

Overview

HTTP API lets departments send across SMS messages using HTTP URL interface. The API supports SMS push (Single SMS and Bulk SMS) and SMS Scheduling.

Terms and Definitions

Sender ID: Sender ID or CLI (Caller Line Identification) is limited to 8 characters in the API. According to TRAI regulations, there will be a 2 character prefix when delivered to the phone. For example if you are passing the SenderID as "cdac_mum", you'll may the SMS delivered as AD-cdac_mum or TA-cdac_mum according the route SMS Gateway chooses.

Message Length: For standard character set 160 characters per SMS is supported. If a message is sent, whose length is longer than permitted characters limit, it shall be broken into multiple messages. You can submit up to 480 characters in one API request.

Using the HTTP URL for sending messages:

The end point of the service is https://msdgweb.mgov.gov.in/esms/sendsmsrequestDLT. This Service is only available on HTTP POST. The Push SMS is only for termination capability on reasonable efforts basis on all available mobile networks in India only, both GSM & CDMA. C-DAC has no control on delivery rate and that it varies based on the response of telecom networks of the operators. Therefore, no assurances are made by the C-DAC in respect of delivery rate.

Department shall use the PUSH Services for sending messages that are transactional in nature and shall make sure that no promotional /Commercial SMSes is sent to a telecom subscriber using C-DAC SMS Service.. Department shall make reasonable efforts not use the C-DAC's Short Message Service connectivity for transmitting SMS's which are obscene, abusive, offensive, unlawful, illegal, sensitive in nature, communal, unauthorized, or compromising the National Security etc.

PUSH Account Creation

Please provide the following details for Push SMS account creation on the MSDG:

Parameter	Description
Organization/Department Name	Name of the Organization /Department
UserName	For login to MSDG Portal (Use alphabets and numbers only. in of 6 &Max of 15 chars)
Password	For login password (Use alphabets and numbers only. Min of 6 &max of 10 chars)
Contact Person Name	Details of the Contact Person
Address	Address of the Department
Mobile Number	For verification and alert messages
Alternate Mobile Number	For verification and alert messages
Email-ID	*For verification and alert mails*
Project Details	Details of the project or services
Sender ID	(maximum up to 6 characters)

PUSH Parameter Definitions:

Following parameters has to be passed along with the SMS Push request.

Parameter	Description
username	Specify the username as given at the time of account creation
password	Password should be encrypted as per below code.
message	The SMS Text you want to submit
numbers	The set of mobile numbers to broadcast the above SMS content. You can pass 10 or 12 digit mobile numbers in comma separated format.Eg : 895123456,9847123456,919809123456
senderid	Sender id should have 6 characters only and only alphabets are allowed no numbers or special characters, all should be uppercase as per new TRAI regulations
key	Optional
messages	This is for using sendpairedsms method in webservice <message> <text> Test Message </text> <numbers>91xxxxxxxxxx</numbers> <?message>
hashValue	Hash value generated from username, senderid, message and securekey
templateid	Templateid (12 or 19 digit) is unique for each message content

API Response Codes

Response Code	Meaning
401	Credentials Error, may be invalid username or password
402, X	X messages submitted successfully
403	Credits not available
404	Internal Database Error
405	Internal Networking Error
406	Invalid or Duplicate numbers
407, 408	Network Error on SMSC
409	SMSC response timed out, message will be submitted
410	Internal Limit Exceeded, Contact support
411, 412	Sender ID not approved.
413	Suspect Spam, we do not accept these messages.
414	Rejected by various reasons by the operator such as DND, SPAM etc
415	Secure Key not available
416	Hash doesn't match
417	otpmsg method can contain at max two mobile numbers
418	Daily Limit Exceeded
422	Entity id should be 12 to 19 digit,

423	TEMPLATE_ID is mandatory and it should be 12 or 19 digit length
425	Secure Key is a mandatory field

PUSH SMS Integration

Java Example for secure push

SMSHTTPSPostClient.java

```
/**
 *
 */
package in.gov.mgov;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.security.KeyManagementException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.List;
import java.util.StringTokenizer;

import javax.net.ssl.SSLContext;

import javax.net.ssl.SSLContext;

import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.scheme.Scheme;
import org.apache.http.conn.ssl.SSLSocketFactory;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;

import org.apache.commons.httpclient.HttpException;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.scheme.Scheme;
import org.apache.http.conn.ssl.SSLSocketFactory;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;

/**
 * @author      Mobile Seva < msdp@cdac.in >
 * <p>Kindly add require Jar files to run Java client</p><p>
 * Apache commons-codec-1.9
 * <br>Apache commons-httpclient-3.1
 * <br>Apache commons-logging-1.2
 * @see <a href="https://mgov.gov.in/doc/RequiredJars.zip">Download required
Jar files</a>
 */
```

```
public class SMSServices {

	/**
	 * Send Single text SMS
	 * @param username : Department Login User Name
	 * @param password : Department Login Password
	 * @param message  : Message e.g. 'Welcome to mobile Seva'
	 * @param senderId      : Department allocated SenderID
	 * @param mobileNumber : Single Mobile Number e.g. '99XXXXXXX'
	 * @param secureKey :  Department key generated by login to services
portal
	 * @param templateid : TEMPLATE_ID is mandatory and it should be 12 or 19
	   digit length
	 * @return {@link String} response from Mobile Seva Gateway e.g.
'402,MsgID = 150620161466003974245msdgsms'
	 * @see <a href="https://mgov.gov.in/msdp_sms_push.jsp">Return types
code details</a>
	 *
	 */

	public String sendSingleSMS(String username, String password , String
message , String senderId, String mobileNumber,String secureKey,String
templateid){


		String responseString = "";
		SSLSocketFactory sf=null;
		SSLContext context=null;
		String encryptedPassword;
		try {
			context=SSLContext.getInstance("TLSv1.2");
			context.init(null, null, null);
			sf=new SSLSocketFactory(context,
SSLSocketFactory.STRICT_HOSTNAME_VERIFIER);
			Scheme scheme=new Scheme("https",443,sf);
			HttpClient client=new DefaultHttpClient();

	client.getConnectionManager().getSchemeRegistry().register(scheme);
			HttpPost post=new
HttpPost("https://msdgweb.mgov.gov.in/esms/sendsmsrequestDLT");
			encryptedPassword  = MD5(password);
			message = message.trim();
			String genratedhashKey = hashGenerator(username, senderId,
message, secureKey);
			List<NameValuePair> nameValuePairs=new
ArrayList<NameValuePair>(1);
			nameValuePairs.add(new BasicNameValuePair("mobileno",
mobileNumber));
			nameValuePairs.add(new BasicNameValuePair("senderid",
senderId));
			nameValuePairs.add(new BasicNameValuePair("content",
message));
			nameValuePairs.add(new BasicNameValuePair("smsservicetype",
"singlemsg"));
			nameValuePairs.add(new BasicNameValuePair("username",
username));
			nameValuePairs.add(new   BasicNameValuePair("password",
encryptedPassword));
			nameValuePairs.add(new BasicNameValuePair("key",
genratedhashKey));
```

```
                    nameValuePairs.add(new BasicNameValuePair("templateid",
templateid));
```

```
post.setEntity(new UrlEncodedFormEntity(nameValuePairs)); HttpResponse
response=client.execute(post); BufferedReader bf=new BufferedReader(new
  InputStreamReader(response.getEntity().getContent()));
                    String line="";
                    while((line=bf.readLine())!=null){
                          responseString = responseString+line;

                    }
                    System.out.println(responseString);
              } catch (NoSuchAlgorithmException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (KeyManagementException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (UnsupportedEncodingException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (ClientProtocolException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              }
              return responseString;
        }

        /**
         * Send Bulk text SMS
         * @param username : Department Login User Name
         * @param password : Department Login Password
         * @param message  : Message e.g. 'Welcome to mobile Seva'
         * @param senderId      : Department allocated SenderID
         * @param mobileNumber : Bulk Mobile Number with comma separated e.g.
  '99XXXXXXX,99XXXXXXXX'
         * @param secureKey :  Department key generated by login to services
  portal
         * @param templateid : TEMPLATE_ID is mandatory and it should be 12 or 19
           digit length
         * @return {@link String} response from Mobile Seva Gateway e.g.
  '402,MsgID = 150620161466003974245msdgsms'
         * @see <a href="https://mgov.gov.in/msdp_sms_push.jsp">Return types
  code details</a>
         *
         */
        public String sendBulkSMS(String username, String password , String
  message , String senderId, String mobileNumber, String secureKey, String
  templateid){

              String responseString = "";
              SSLSocketFactory sf=null;
              SSLContext context=null;
              String encryptedPassword;
              try {
                    context=SSLContext.getInstance("TLSv1.2");
                    context.init(null, null, null);
                    sf=new SSLSocketFactory(context,
  SSLSocketFactory.STRICT_HOSTNAME_VERIFIER);
```

```
                    Scheme scheme=new Scheme("https",443,sf);
                    HttpClient client=new DefaultHttpClient();

      client.getConnectionManager().getSchemeRegistry().register(scheme);
                    HttpPost post=new
HttpPost("https://msdgweb.mgov.gov.in/esms/sendsmsrequestDLT");
                    encryptedPassword  = MD5(password);
                    message = message.trim();
                    String genratedhashKey = hashGenerator(username, senderId,
message, secureKey);
                    List<NameValuePair> nameValuePairs=new
ArrayList<NameValuePair>(1);
                    nameValuePairs.add(new BasicNameValuePair("bulkmobno",
mobileNumber));
                    nameValuePairs.add(new BasicNameValuePair("senderid",
senderId));
                    nameValuePairs.add(new BasicNameValuePair("content",
message));
                    nameValuePairs.add(new BasicNameValuePair("smsservicetype",
"bulkmsg"));
                    nameValuePairs.add(new BasicNameValuePair("username",
username));
                    nameValuePairs.add(new   BasicNameValuePair("password",
encryptedPassword));
                    nameValuePairs.add(new BasicNameValuePair("key",
genratedhashKey));
                    nameValuePairs.add(new BasicNameValuePair("templateid",
                    templateid));
                    post.setEntity(new UrlEncodedFormEntity(nameValuePairs));
                    HttpResponse response=client.execute(post);
                    BufferedReader bf=new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
                    String line="";
                    while((line=bf.readLine())!=null){
                          responseString = responseString+line;

                    }
                    System.out.println(responseString);
              } catch (NoSuchAlgorithmException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (KeyManagementException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (UnsupportedEncodingException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (ClientProtocolException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              }
              return responseString;
        }
        /**
         * Send Unicode text SMS
         * @param username : Department Login User Name
```

```
        * @param password : Department Login Password
        * @param message  : Unicode Message e.g. 'à¤µà¤¿à¤•à¤¾à¤¸
        * @param senderId      : Department allocated SenderID
        * @param mobileNumber : Bulk Mobile Number with comma separated e.g.
'99XXXXXXX,99XXXXXXXX'
        * @param secureKey :  Department key generated by login to services
portal
         @param templateid : TEMPLATE_ID is mandatory and it should be 12 or 19
       digit length
        * @return {@link String} response from Mobile Seva Gateway e.g.
'402,MsgID = 150620161466003974245msdgsms'
        * @see <a href="https://mgov.gov.in/msdp_sms_push.jsp">Return types
code details</a>
        *
        */
       public String sendUnicodeSMS(String username, String password , String
message , String senderId, String mobileNumber,String secureKey, String
templateid){


            String finalmessage = "";
            message =message.trim();
            for(int i = 0 ; i< message.length();i++){

                  char ch = message.charAt(i);
                  int j = (int) ch;
                  String sss = "&#"+j+";";
                  finalmessage =finalmessage+sss;
            }



            String responseString = "";
            SSLSocketFactory sf=null;
            SSLContext context=null;
            String encryptedPassword;
            try {
                  context=SSLContext.getInstance("TLSv1.2");
                  context.init(null, null, null);
                  sf=new SSLSocketFactory(context,
SSLSocketFactory.STRICT_HOSTNAME_VERIFIER);
                  Scheme scheme=new Scheme("https",443,sf);
                  HttpClient client=new DefaultHttpClient();

      client.getConnectionManager().getSchemeRegistry().register(scheme);
                  HttpPost post=new
HttpPost("https://msdgweb.mgov.gov.in/esms/sendsmsrequestDLT");
                  encryptedPassword  = MD5(password);
                  String genratedhashKey = hashGenerator(username, senderId,
finalmessage, secureKey);
                  List<NameValuePair> nameValuePairs=new
ArrayList<NameValuePair>(1);
                  nameValuePairs.add(new BasicNameValuePair("bulkmobno",
mobileNumber));

senderId));

finalmessage));

"unicodemsg"));
```

```
nameValuePairs.
add(new
BasicNameValueP
air("senderid",
nameValuePairs.
add(new
BasicNameValueP
air("content",
nameValuePairs.
add(new
BasicNameValueP
air("smsservice
type",
```

```
                    nameValuePairs.add(new   BasicNameValuePair("username",
username));
                    nameValuePairs.add(new   BasicNameValuePair("password",
encryptedPassword));
                    nameValuePairs.add(new BasicNameValuePair("key",
genratedhashKey));
                     nameValuePairs.add(new BasicNameValuePair("templateid",
                     templateid));
                    post.setEntity(new UrlEncodedFormEntity(nameValuePairs));
                    HttpResponse response=client.execute(post);
                    BufferedReader bf=new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
                    String line="";
                    while((line=bf.readLine())!=null){
                          responseString = responseString+line;

                    }
                    System.out.println(responseString);
              } catch (NoSuchAlgorithmException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (KeyManagementException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (UnsupportedEncodingException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (ClientProtocolException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              }
              return responseString;
        }

        /**
         * Send Single OTP text SMS

*</namevaluepair></namevaluepair></namevaluepair></namevaluepair></namevaluep
air></namevaluepair></p><p> Use only in case of OTP related message</p><p>
         * Messages other than OTP will not be delivered to the users
         * @param username : Department Login User Name
         * @param password : Department Login Password
         * @param message  : Message e.g. 'Welcome to mobile Seva'
         * @param senderId      : Department allocated SenderID
         * @param mobileNumber : Single Mobile Number e.g. '99XXXXXXX'
         * @param secureKey :  Department key generated by login to services
portal
           @param templateid : TEMPLATE_ID is mandatory and it should be 12 or 19
        digit length
         * @return {@link String} response from Mobile Seva Gateway e.g.
'402,MsgID = 150620161466003974245msdgsms'
         * @see <a href="https://mgov.gov.in/msdp_sms_push.jsp">Return types
code details</a>
         *
         */
```

```
public String sendOtpSMS(String username, String password , String
message , String senderId, String mobileNumber,String secureKey, String
templateid){
```

```
		String responseString = "";
		SSLSocketFactory sf=null;
		SSLContext context=null;
		String encryptedPassword;
		try {
			context=SSLContext.getInstance("TLSv1.2");
			context.init(null, null, null);
			sf=new SSLSocketFactory(context,
SSLSocketFactory.STRICT_HOSTNAME_VERIFIER);
			Scheme scheme=new Scheme("https",443,sf);
			HttpClient client=new DefaultHttpClient();

	client.getConnectionManager().getSchemeRegistry().register(scheme);
			HttpPost post=new
HttpPost("https://msdgweb.mgov.gov.in/esms/sendsmsrequestDLT");
			encryptedPassword  = MD5(password);
			message = message.trim();
			String genratedhashKey = hashGenerator(username, senderId,
message, secureKey);
			List<NameValuePair> nameValuePairs=new
ArrayList<NameValuePair>(1);
			nameValuePairs.add(new BasicNameValuePair("mobileno",
mobileNumber));
			nameValuePairs.add(new BasicNameValuePair("senderid",
senderId));
			nameValuePairs.add(new BasicNameValuePair("content",
message));
			nameValuePairs.add(new BasicNameValuePair("smsservicetype",
"otpmsg"));
			nameValuePairs.add(new BasicNameValuePair("username",
username));
			nameValuePairs.add(new   BasicNameValuePair("password",
encryptedPassword));
			nameValuePairs.add(new BasicNameValuePair("key",
genratedhashKey));
			 nameValuePairs.add(new BasicNameValuePair("templateid",
			 templateid));
			post.setEntity(new UrlEncodedFormEntity(nameValuePairs));
			HttpResponse response=client.execute(post);
			BufferedReader bf=new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
			String line="";
			while((line=bf.readLine())!=null){
				responseString = responseString+line;

			}
			System.out.println(responseString);
		} catch (NoSuchAlgorithmException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (KeyManagementException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (UnsupportedEncodingException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (ClientProtocolException e) {
			// TODO Auto-generated catch block
```

```
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return responseString;

        }

        /**
         * Send Single Unicode OTP text SMS
         * @param username : Department Login User Name
         * @param password : Department Login Password
         * @param message  : Unicode Message e.g. 'à¤µà¤¿à¤•à¤¾à¤¸ à¤†à¤£à¤¿
à¤ªà¥•à¤°à¤—à¤¤ à¤¸à¤‚à¤—à¤£à¤¨ à¤•à¥‡à¤‚,à¤¦à¥•à¤° à¤®à¤§à¥□à¤¯ à¤¸à¥‡
à¤¸à¥•à¤µà¤¾à¤—à¤¤ à¤†à¤¹à¥‡'
         * @param senderId      : Department allocated SenderID
         * @param mobileNumber : Bulk Mobile Number with comma separated e.g.
'99XXXXXXX,99XXXXXXXX'
         * @param secureKey :  Department key generated by login to services
portal
          @param templateid : TEMPLATE_ID is mandatory and it should be 12 or 19
        digit length
         * @return {@link String} response from Mobile Seva Gateway e.g.
'402,MsgID = 150620161466003974245msdgsms'
         * @see <a href="https://mgov.gov.in/msdp_sms_push.jsp">Return types
code details</a>
         *
         */
        public String sendUnicodeOtpSMS(String username, String password ,
                String message , String senderId, String mobileNumber,String
                                                secureKey,String templateid){

                message =message.trim();
                String finalmessage = "";
                for(int i = 0 ; i< message.length();i++){

                        char ch = message.charAt(i);
                        int j = (int) ch;
                        String sss = "&#"+j+";";
                        finalmessage =finalmessage+sss;
                }




                String responseString = "";
                SSLSocketFactory sf=null;
                SSLContext context=null;
                String encryptedPassword;
                try {
                        context=SSLContext.getInstance("TLSv1.2");
                        context.init(null, null, null);
                        sf=new SSLSocketFactory(context,
SSLSocketFactory.STRICT_HOSTNAME_VERIFIER);
                        Scheme scheme=new Scheme("https",443,sf);
                        HttpClient client=new DefaultHttpClient();

        client.getConnectionManager().getSchemeRegistry().register(scheme);
```

```
                    HttpPost post=new
HttpPost("https://msdgweb.mgov.gov.in/esms/sendsmsrequestDLT");
                    encryptedPassword  = MD5(password);
                    String genratedhashKey = hashGenerator(username, senderId,
finalmessage, secureKey);
                    List<NameValuePair> nameValuePairs=new
ArrayList<NameValuePair>(1);
                    nameValuePairs.add(new BasicNameValuePair("mobileno",
mobileNumber));
                    nameValuePairs.add(new BasicNameValuePair("senderid",
senderId));
                    nameValuePairs.add(new BasicNameValuePair("content",
finalmessage));
                    nameValuePairs.add(new BasicNameValuePair("smsservicetype",
"unicodeotpmsg"));
                    nameValuePairs.add(new BasicNameValuePair("username",
username));
                    nameValuePairs.add(new   BasicNameValuePair("password",
encryptedPassword));
                    nameValuePairs.add(new BasicNameValuePair("key",
genratedhashKey));
                     nameValuePairs.add(new BasicNameValuePair("templateid",
                     templateid));
                    post.setEntity(new UrlEncodedFormEntity(nameValuePairs));
                    HttpResponse response=client.execute(post);
                    BufferedReader bf=new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
                    String line="";
                    while((line=bf.readLine())!=null){
                          responseString = responseString+line;

                    }
                    System.out.println(responseString);
              } catch (NoSuchAlgorithmException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (KeyManagementException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (UnsupportedEncodingException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (ClientProtocolException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              }
              return responseString;
                    }

      protected String hashGenerator(String userName, String senderId, String
content, String secureKey) {
              // TODO Auto-generated method stub
              StringBuffer finalString=new StringBuffer();

      finalString.append(userName.trim()).append(senderId.trim()).append(cont
ent.trim()).append(secureKey.trim());
```

```
		//		logger.info("Parameters for SHA-512 : "+finalString);
		String hashGen=finalString.toString();
		StringBuffer sb = null;
		MessageDigest md;
		try {
			md = MessageDigest.getInstance("SHA-512");
			md.update(hashGen.getBytes());
			byte byteData[] = md.digest();
			//convert the byte to hex format method 1
			sb = new StringBuffer();
			for (int i = 0; i < byteData.length; i++) {
				sb.append(Integer.toString((byteData[i] & 0xff) +
0x100, 16).substring(1));
			}

		} catch (NoSuchAlgorithmException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		return sb.toString();
	}

	/**
	 * Get units of the unicode message
	 * @param message e.g. 'ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Âµ
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¿
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã‚Â¢â‚¬Å¡Ã¢â□šÃ¬Ãƒâ□šÃ‚Â¢
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¾
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¸ ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã‚Â¢Ãƒâ□šÃ¢â□šÃ¬ ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â£ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¿ ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¥ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¿Â½ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â° ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã‚Â¢Ãƒâ□šÃ¢â□šÃ¬Ãƒâ□šÃ¢â‚¬Å¡Ã¬ÃƒÆ'Ã†â□
™ ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¸ ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¤ÃƒÆ'Ã‚Â¢Ãƒâ□šÃ¢â□šÃ¬Ãƒâ□šÃ…Â¡ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã‚Â¢Ãƒâ□šÃ¢â□šÃ¬Ãƒâ□šÃ¢â‚¬Å¡Ã¬ÃƒÆ'Ã†â□
™ ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â£ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¨ ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã‚Â¢Ãƒâ□šÃ¢â□šÃ¬Ãƒâ□šÃ‚Â¢ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¥ÃƒÆ'Ã‚Â¢Ãƒâ□šÃ¢â□šÃ¬Ãƒâ□šÃ‚Â¡ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¡ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¥ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¿Â½ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â° ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â®ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â§ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¥ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¿Â½ÃƒÆ'Ã†â□™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ□šÃ‚Â ÃƒÆ'Ã†â□™
```

ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¥ÃƒÆ'Ã‚Â¢Ãƒâ€šÃ¢â€šÂ¬Ãƒâ€šÃ‚Â¡ ÃƒÆ'Ã†â€™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¸ÃƒÆ'Ã†â€™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¥ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¿Â½ÃƒÆ'Ã†â€™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¼ÃƒÆ'Ã†â€™

```
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¾ÃƒÆ'Ã†â€™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¤ÃƒÆ'Ã‚Â¢Ã¢â‚¬Å¡Ã‚Â¬Ã…Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¬Ã¯¿½ÃƒÆ'Ã†â€™
â„¢ ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¤ ÃƒÆ'Ã†â€™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¤ÃƒÆ'Ã‚Â¢Ã¢â‚¬Å¡Ã‚Â¬Ã…Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¬ÃƒÆ'Ã†â€™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¤ÃƒÆ'Ã¢â‚¬Å¡Ã‚Â¬Ã…Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¥ÃƒÆ'Ã†â€™
ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¥ÃƒÆ'Ã‚Â¢Ã¢â‚¬Å¡Ã‚Â¬Ã…Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¬ÃƒÆ'Ã¢â‚¬Å¡Ãƒâ€šÃ‚Â¡'
        * @return int message unit
        * **/
       public int getUnicodeTextMessageUnit(String message) {
             String charInUnicode = "";
             int msgUnit = 1;
             int msgLen = 0;
             String unicodeMessgae = "";
             String finalmessage = null;
             for(int i = 0 ; i < message.length();i++){

                   char ch = message.charAt(i);
                   int j = (int) ch;
                   String sss = "&#"+j+";";
                   finalmessage =finalmessage+sss;
             }
             StringTokenizer st = new StringTokenizer(finalmessage, " ");
             while (st.hasMoreElements()) {
                   String str1 = (String) st.nextElement();
                   StringTokenizer dd = new StringTokenizer(str1, ";");
                   while (dd.hasMoreElements()) {
                         charInUnicode = (String) dd.nextElement();
                        if (charInUnicode.startsWith("&#")) {
                              StringTokenizer df = new StringTokenizer(
                                          charInUnicode, "&#");
                              while (df.hasMoreElements()) {
                                    String kk = (String) df.nextElement();
                                    unicodeMessgae = unicodeMessgae + "," +
kk;
                                    msgLen = msgLen+1;
                              }

                        } else {
                              if(charInUnicode.contains("&#")){
                                    StringTokenizer st1 =new
StringTokenizer(charInUnicode, "&#");
                                    while (st1.hasMoreElements()) {
                                          String kk = (String)
st1.nextElement();
                                          for (int i1 = 0; i1 < kk.length();
++i1) {

                                                char c =kk.charAt(i1);
                                                int j = (int) c;
                                                unicodeMessgae =
unicodeMessgae + "," + j;

                                                msgLen = msgLen+1;
                                          }
                                          String uni = st1.nextToken();
                                          unicodeMessgae = unicodeMessgae +
"," + uni;
                                          msgLen = msgLen+1;
```

```
                                        }
                                  }

                                  else{
                                        for (int i1 = 0; i1 <
 charInUnicode.length(); ++i1) {
                                              char c = charInUnicode.charAt(i1);
                                              int j = (int) c;
                                              unicodeMessgae = unicodeMessgae +
","  + j;
                                              msgLen = msgLen+1;
                                        }
                                  }
                            }

                      }
                      unicodeMessgae = unicodeMessgae + " ";
               }

              if (msgLen > 70) {

                      msgUnit = 2;

                     if (msgLen > 134) {
                            msgUnit = 3;
                            if (msgLen > 201) {
                                  msgUnit = 4;
                                  if (msgLen > 268) {
                                        msgUnit = 5;
                                        if (msgLen > 335) {
                                              msgUnit = 6;
                                              if (msgLen > 402) {
                                                    msgUnit = 7;
                                                    if (msgLen> 469) {
                                                          msgUnit = 8;
                                                          if (msgLen > 536) {
                                                                msgUnit = 9;
                                                                if (msgLen > 603)
{
                                                                      msgUnit =
10;

                                                          }
                                                    }
                                              }
                                        }
                                  }
                            }
                      }

               }else{
                      msgUnit = 1;
               }
               return msgUnit;
```

```
	}

	/**
	 * Get units of the text message
	 * @param message e.g. 'Welcome to Mobile Seva'
	 * @return int message unit
	 * **/
	public int getNormalTextMessageUnit(String message) {

		int msgUnit = 1;
		if (message.length() > 160) {
			msgUnit = 2;
			if (message.length() > 306) {
				msgUnit = 3;
			}
			if (message.length() > 459) {
				msgUnit = 4;
			}
			if (message.length() > 612) {
				msgUnit = 5;
			}
			if (message.length() > 765) {
				msgUnit = 6;
			}
			if (message.length() > 918) {
				msgUnit = 7;
			}
			if (message.length() > 1071) {
				msgUnit = 8;
			}
			if (message.length() > 1224) {
				msgUnit = 9;
			}
			if (message.length() > 1377) {
				msgUnit = 10;
			}
		}else{
			msgUnit = 1;
		}
		return msgUnit;
	}


	/****
	 * Method  to convert Normal Plain Text Password to MD5 encrypted
password
	 ***/

	private static String MD5(String text) throws NoSuchAlgorithmException,
UnsupportedEncodingException
	{
		MessageDigest md;
		md = MessageDigest.getInstance("SHA-1");
		byte[] md5 = new byte[64];
		md.update(text.getBytes("iso-8859-1"), 0, text.length());
		md5 = md.digest();
```

```
            return convertedToHex(md5);
        }

        private static String convertedToHex(byte[] data)
        {
            StringBuffer buf = new StringBuffer();

            for (int i = 0; i < data.length; i++)
            {
                int halfOfByte = (data[i] >>> 4) & 0x0F;
                int twoHalfBytes = 0;

                do
                {
                    if ((0 <= halfOfByte) && (halfOfByte <= 9))
                    {
                        buf.append( (char) ('0' + halfOfByte) );
                    }

                    else
                    {
                        buf.append( (char) ('a' + (halfOfByte - 10)) );
                    }

                    halfOfByte = data[i] & 0x0F;

                } while(twoHalfBytes++ < 1);
            }
            return buf.toString();
        }

}
```

.NET (C#) Example for secure push:

SMSHTTPSPostClient.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net;
using System.Web;
using System.IO;

using System.Security.Cryptography;

using System.Security.Cryptography.X509Certificates;

namespace esms_client
{

    public class SMSHttpPostClient
    {

        /// <summary>
        /// Method for sending single SMS.
        /// </summary>
        /// <param name="username"> Registered user name</param>
        /// <param name="password"> Valid login password</param>
        /// <param name="senderid">Sender ID</param>
        /// <param name="mobileNo"> valid Single Mobile Number </param>
        /// <param name="message">Message Content</param>
        /// <param name="secureKey">Department generate key by login to
services portal
        /// <param name="templateid">Templateid is unique for each message
content</param>


        // Method for sending single SMS.

        public String sendSingleSMS(String username, String password, String
senderid, String mobileNo, String message, String secureKey, String templateid)

        {
            //Latest Generated Secure Key Stream dataStream;
            System.Net.ServicePointManager.SecurityProtocol =
SecurityProtocolType.Tls12 | SecurityProtocolType.Tls11 |
SecurityProtocolType.Tls;
            HttpWebRequest request =
(HttpWebRequest)WebRequest.Create("https://msdgweb.mgov.gov.in/esms/sendsmsre
questDLT");
            request.ProtocolVersion = HttpVersion.Version10;
            request.KeepAlive = false;
            request.ServicePoint.ConnectionLimit = 1;
            //((HttpWebRequest)request).UserAgent = ".NET Framework Example
Client";
```

```
            ((HttpWebRequest)request).UserAgent = "Mozilla/4.0 (compatible;
MSIE 5.0; Windows 98; DigExt)";

            request.Method = "POST";
            System.Net.ServicePointManager.CertificatePolicy = new
MyPolicy();

            String encryptedPassword = encryptedPasswod(password);
            String NewsecureKey = hashGenerator(username.Trim(),
senderid.Trim(), message.Trim(), secureKey.Trim());
            String smsservicetype = "singlemsg"; //For single message.

            String query = "username=" +
HttpUtility.UrlEncode(username.Trim())   +
                "&password=" + HttpUtility.UrlEncode(encryptedPassword) +

                "&smsservicetype=" + HttpUtility.UrlEncode(smsservicetype) +

                "&content=" + HttpUtility.UrlEncode(message.Trim()) +

                "&mobileno=" + HttpUtility.UrlEncode(mobileNo) +

                "&senderid=" + HttpUtility.UrlEncode(senderid.Trim()) +
              "&key=" + HttpUtility.UrlEncode(NewsecureKey.Trim())+
              "&templateid=" +HttpUtility.UrlEncode(templateid.Trim());



            byte[] byteArray = Encoding.ASCII.GetBytes(query);

            request.ContentType = "application/x-www-form-urlencoded";

            request.ContentLength = byteArray.Length;


            dataStream = request.GetRequestStream();

            dataStream.Write(byteArray, 0, byteArray.Length);

            dataStream.Close();

            WebResponse response = request.GetResponse();

            String Status = ((HttpWebResponse)response).StatusDescription;

            dataStream = response.GetResponseStream();

            StreamReader reader = new StreamReader(dataStream);

            String responseFromServer = reader.ReadToEnd();

            reader.Close();

            dataStream.Close();

            response.Close();
```

```
            return responseFromServer;

        }


        /// <summary>
        /// Method for sending bulk SMS.
        /// </summary>
        /// <param name="username"> Registered user name</param>
        /// <param name="password"> Valid login password</param>
        /// <param name="senderid">Sender ID</param>
        /// <param name="mobileNo"> valid Mobile Numbers </param>
        /// <param name="message">Message Content</param>
        /// <param name="secureKey">Department generate key by login to
services portal
        /// <param name="templateid">Templateid is unique for each message
content </param>

        // method for sending bulk SMS

        public String sendBulkSMS(String username, String password, String
senderid, String mobileNos, String message,String secureKey,String templateid)

        {
            Stream dataStream;
            System.Net.ServicePointManager.SecurityProtocol =
SecurityProtocolType.Tls12 | SecurityProtocolType.Tls11 |
SecurityProtocolType.Tls;
            HttpWebRequest request =
(HttpWebRequest)WebRequest.Create("https://msdgweb.mgov.gov.in/esms/sendsmsre
questDLT");
            request.ProtocolVersion = HttpVersion.Version10;
            request.KeepAlive = false;
            request.ServicePoint.ConnectionLimit = 1;
            //((HttpWebRequest)request).UserAgent = ".NET Framework Example
Client";
            ((HttpWebRequest)request).UserAgent = "Mozilla/4.0 (compatible;
MSIE 5.0; Windows 98; DigExt)";

            request.Method = "POST";
            System.Net.ServicePointManager.CertificatePolicy = new
MyPolicy();

            String encryptedPassword = encryptedPasswod(password);
            String NewsecureKey = hashGenerator(username.Trim(),
senderid.Trim(), message.Trim(), secureKey.Trim());
            Console.Write(NewsecureKey);
            Console.Write(encryptedPassword);

            String smsservicetype = "bulkmsg"; // for bulk msg

            String query = "username=" +
HttpUtility.UrlEncode(username.Trim())  +

             "&password=" + HttpUtility.UrlEncode(encryptedPassword) +

             "&smsservicetype=" + HttpUtility.UrlEncode(smsservicetype) +
```

```
                "&content=" + HttpUtility.UrlEncode(message.Trim()) +

                "&bulkmobno=" + HttpUtility.UrlEncode(mobileNos) +

                "&senderid=" + HttpUtility.UrlEncode(senderid.Trim()) +

               "&key=" + HttpUtility.UrlEncode(NewsecureKey.Trim())+
               "&templateid=" + HttpUtility.UrlEncode(templateid.Trim());

                Console.Write(query);

                byte[] byteArray = Encoding.ASCII.GetBytes(query);

               request.ContentType = "application/x-www-form-urlencoded";

                request.ContentLength = byteArray.Length;

                dataStream = request.GetRequestStream();

                dataStream.Write(byteArray, 0, byteArray.Length);

                dataStream.Close();

                WebResponse response = request.GetResponse();

                String Status = ((HttpWebResponse)response).StatusDescription;

                dataStream = response.GetResponseStream();

                StreamReader reader = new StreamReader(dataStream);

                String responseFromServer = reader.ReadToEnd();

                reader.Close();

                dataStream.Close();

                response.Close();
                return responseFromServer;

            }


            /// <summary>
            /// method for Sending unicode..
            /// </summary>
            /// <param name="username"> Registered user name</param>
            /// <param name="password"> Valid login password</param>
            /// <param name="senderid">Sender ID</param>
            /// <param name="mobileNo"> valid Mobile Numbers </param>
            /// <param name="Unicodemessage">Unicodemessage Message Content
</param>
            /// <param name="secureKey">Department generate key by login to
services portal
/// <param name="templateid">Templateid is unique for each message content
</param>

            //method for Sending unicode message..
```

```
        public String sendUnicodeSMS(String username, String password, String
senderid, String mobileNos, String Unicodemessage, String secureKey, String
templateid)

        {
            Stream dataStream;
            System.Net.ServicePointManager.SecurityProtocol =
SecurityProtocolType.Tls12 | SecurityProtocolType.Tls11 |
SecurityProtocolType.Tls;
            HttpWebRequest request =
(HttpWebRequest)WebRequest.Create("https://msdgweb.mgov.gov.in/esms/sendsmsre
questDLT");
            request.ProtocolVersion = HttpVersion.Version10;
            request.KeepAlive = false;
            request.ServicePoint.ConnectionLimit = 1;
            //((HttpWebRequest)request).UserAgent = ".NET Framework Example
Client";
            ((HttpWebRequest)request).UserAgent = "Mozilla/4.0 (compatible;
MSIE 5.0; Windows 98; DigExt)";

            request.Method = "POST";
            System.Net.ServicePointManager.CertificatePolicy = new
MyPolicy();
            String U_Convertedmessage = "";

            foreach (char c in Unicodemessage)
            {
                int j = (int)c;
               String sss = "&#" + j + ";";
                U_Convertedmessage = U_Convertedmessage +sss;
            }
            String encryptedPassword = encryptedPasswod(password);
            String NewsecureKey = hashGenerator(username.Trim(),
senderid.Trim(), U_Convertedmessage.Trim(), secureKey.Trim());


            String smsservicetype = "unicodemsg"; // for unicode msg
            String query = "username=" +
HttpUtility.UrlEncode(username.Trim())  +
                "&password=" + HttpUtility.UrlEncode(encryptedPassword) +
                "&smsservicetype=" + HttpUtility.UrlEncode(smsservicetype) +
                "&content=" +
HttpUtility.UrlEncode(U_Convertedmessage.Trim()) +
                "&bulkmobno=" + HttpUtility.UrlEncode(mobileNos) +
                "&senderid=" + HttpUtility.UrlEncode(senderid.Trim()) +
                "&key=" +HttpUtility.UrlEncode(NewsecureKey.Trim())+
                "&templateid=" +HttpUtility.UrlEncode(templateid.Trim());



            byte[] byteArray = Encoding.ASCII.GetBytes(query);
            request.ContentType = "application/x-www-form-urlencoded";
            request.ContentLength = byteArray.Length;
            dataStream = request.GetRequestStream();
            dataStream.Write(byteArray, 0, byteArray.Length);
            dataStream.Close();
            WebResponse response = request.GetResponse();
            String Status = ((HttpWebResponse)response).StatusDescription;
```

```
            dataStream = response.GetResponseStream();
            StreamReader reader = new StreamReader(dataStream);
            String responseFromServer = reader.ReadToEnd();
            reader.Close();
            dataStream.Close();
            response.Close();
            return responseFromServer;
        }



        /// <summary>
        /// Method for sending OTP MSG.
        /// </summary>
        /// <param name="username"> Registered user name</param>
        /// <param name="password"> Valid login password</param>
        /// <param name="senderid">Sender ID</param>
        /// <param name="mobileNo"> valid single  Mobile Number </param>
        /// <param name="message">Message Content</param>
        /// <param name="secureKey">Department generate key by login to
services portal
        /// <param name="templateid">Templateid is unique for each message
content </param>

        // Method for sending OTP MSG.

        public String sendOTPMSG(String username, String password, String
senderid, String mobileNo, String message, String secureKey, String templateid)

        {
            Stream dataStream;
            System.Net.ServicePointManager.SecurityProtocol =
SecurityProtocolType.Tls12 | SecurityProtocolType.Tls11 |
SecurityProtocolType.Tls;
            HttpWebRequest request =
(HttpWebRequest)WebRequest.Create("https://msdgweb.mgov.gov.in/esms/sendsmsre
questDLT");
            request.ProtocolVersion = HttpVersion.Version10;
            request.KeepAlive = false;
            request.ServicePoint.ConnectionLimit = 1;
            //((HttpWebRequest)request).UserAgent = ".NET Framework Example
Client";
            ((HttpWebRequest)request).UserAgent = "Mozilla/4.0 (compatible;
MSIE 5.0; Windows 98; DigExt)";
            request.Method = "POST";
            System.Net.ServicePointManager.CertificatePolicy = new
MyPolicy();

            String encryptedPassword = encryptedPasswod(password);
            String key = hashGenerator(username.Trim(), senderid.Trim(),
message.Trim(), secureKey.Trim());

            String smsservicetype = "otpmsg"; //For OTP message.

            String query = "username=" +
HttpUtility.UrlEncode(username.Trim())  +
                "&password=" + HttpUtility.UrlEncode(encryptedPassword) +
```

```
                "&smsservicetype=" + HttpUtility.UrlEncode(smsservicetype) +

                "&content=" + HttpUtility.UrlEncode(message.Trim()) +

                "&mobileno=" + HttpUtility.UrlEncode(mobileNo) +

                "&senderid=" + HttpUtility.UrlEncode(senderid.Trim()) +
              "&key=" + HttpUtility.UrlEncode(key.Trim())+
                "&templateid=" +HttpUtility.UrlEncode(templateid.Trim());



            byte[] byteArray = Encoding.ASCII.GetBytes(query);

            request.ContentType = "application/x-www-form-urlencoded";

            request.ContentLength = byteArray.Length;


            dataStream = request.GetRequestStream();

            dataStream.Write(byteArray, 0, byteArray.Length);

            dataStream.Close();

            WebResponse response = request.GetResponse();

            String Status = ((HttpWebResponse)response).StatusDescription;

            dataStream = response.GetResponseStream();

            StreamReader reader = new StreamReader(dataStream);

            String responseFromServer = reader.ReadToEnd();

            reader.Close();

            dataStream.Close();

            response.Close();
            return responseFromServer;

        }


        // Method for sending UnicodeOTP MSG.

        /// <summary>
        /// method for Sending unicode..
        /// </summary>
        /// <param name="username"> Registered user name</param>
        /// <param name="password"> Valid login password</param>
        /// <param name="senderid">Sender ID</param>
        /// <param name="mobileNo"> valid Mobile Numbers </param>
        /// <param name="Unicodemessage">Unicodemessage Message Content
</param>
```

```
        /// <param name="secureKey">Department generate key by login to
services portal
        /// <param name="templateid">Templateid is unique for each message
content </param>

        //method for Sending unicode message..

        public String sendUnicodeOTPSMS(String username, String password,
String senderid, String mobileNos, String UnicodeOTPmsg, String secureKey,
String templateid)
        {
            Stream dataStream;
            System.Net.ServicePointManager.SecurityProtocol =
SecurityProtocolType.Tls12 | SecurityProtocolType.Tls11 |
SecurityProtocolType.Tls;
            HttpWebRequest request =
(HttpWebRequest)WebRequest.Create("https://msdgweb.mgov.gov.in/esms/sendsmsre
questDLT");
            request.ProtocolVersion = HttpVersion.Version10;
            request.KeepAlive = false;
            request.ServicePoint.ConnectionLimit = 1;
            //((HttpWebRequest)request).UserAgent = ".NET Framework Example
Client";
            ((HttpWebRequest)request).UserAgent = "Mozilla/4.0 (compatible;
MSIE 5.0; Windows 98; DigExt)";
            request.Method = "POST";
            System.Net.ServicePointManager.CertificatePolicy = new
MyPolicy();

            String U_Convertedmessage = "";

            foreach (char c in UnicodeOTPmsg)
            {
                int j = (int)c;
                String sss = "&#" + j + ";";
                U_Convertedmessage = U_Convertedmessage + sss;
            }
            String encryptedPassword = encryptedPasswod(password);
            String NewsecureKey = hashGenerator(username.Trim(),
senderid.Trim(), U_Convertedmessage.Trim(), secureKey.Trim());


            String smsservicetype = "unicodeotpmsg"; // for unicode msg
            String query = "username=" +
HttpUtility.UrlEncode(username.Trim())  +
                "&password=" + HttpUtility.UrlEncode(encryptedPassword) +
                "&smsservicetype=" + HttpUtility.UrlEncode(smsservicetype) +
                "&content=" +
HttpUtility.UrlEncode(U_Convertedmessage.Trim()) +
                "&bulkmobno=" + HttpUtility.UrlEncode(mobileNos) +
                "&senderid=" + HttpUtility.UrlEncode(senderid.Trim()) +
                "&key=" +HttpUtility.UrlEncode(NewsecureKey.Trim())+
                "&templateid=" +HttpUtility.UrlEncode(templateid.Trim());


            byte[] byteArray = Encoding.ASCII.GetBytes(query);
            request.ContentType = "application/x-www-form-urlencoded";
            request.ContentLength = byteArray.Length;
            dataStream = request.GetRequestStream();
```

```
            dataStream.Write(byteArray, 0, byteArray.Length);
            dataStream.Close();
            WebResponse response = request.GetResponse();
            String Status = ((HttpWebResponse)response).StatusDescription;
            dataStream = response.GetResponseStream();
            StreamReader reader = new StreamReader(dataStream);
            String responseFromServer = reader.ReadToEnd();
            reader.Close();
            dataStream.Close();
            response.Close();
            return responseFromServer;
        }


        /// <summary>
        /// Method to get Encrypted the password
        /// </summary>
        /// <param name="password"> password as String"</param>

        protected String encryptedPasswod(String password)
        {

            byte[] encPwd = Encoding.UTF8.GetBytes(password);
            //static byte[] pwd = new byte[encPwd.Length];
            HashAlgorithm sha1 = HashAlgorithm.Create("SHA1");
            byte[] pp = sha1.ComputeHash(encPwd);
            // static string result =
System.Text.Encoding.UTF8.GetString(pp);
            StringBuilder sb = new StringBuilder();
            foreach (byte b in pp)
            {

                sb.Append(b.ToString("x2"));
            }
            return sb.ToString();

        }

        /// <summary>
        /// Method to Generate hash code
        /// </summary>
        /// <param name= "secure_key">your last generated Secure_key </param>

        protected String hashGenerator(String Username, String sender_id,
String message, String secure_key)
        {

            StringBuilder sb = new StringBuilder();

sb.Append(Username).Append(sender_id).Append(message).Append(secure_key);
            byte[] genkey =Encoding.UTF8.GetBytes(sb.ToString());
            //static byte[] pwd = new byte[encPwd.Length];
            HashAlgorithm sha1 = HashAlgorithm.Create("SHA512");
            byte[] sec_key = sha1.ComputeHash(genkey);

            StringBuilder sb1 = new StringBuilder();
            for (int i = 0; i < sec_key.Length; i++)
```

```
            {
                sb1.Append(sec_key[i].ToString("x2"));
            }
            return sb1.ToString();
        }

    }

}

 class MyPolicy : ICertificatePolicy
        {
            public bool CheckValidationResult(ServicePoint srvPoint,
X509Certificate certificate, WebRequest request, int certificateProblem)
            {
                return true;
            }

        }
```

PHP Example for secure push

SMSHttpPostClient.php

```
<?php
 header('Content-Type: text/html;');
 $username="XXXXXX"; //username of the department
 $password="XXXXXX"; //password of the department
 $senderid="XXXXXX"; //senderid of the deparment
 $message="Your Normal message here "; //message content
 $messageUnicode="?????????????????????? "; //message content in unicode
 $mobileno="86XXXXXX72"; //if single sms need to be send use mobileno keyword
 $mobileNos= "86XXXXXX72,79XXXXXX00"; //if bulk sms need to send use
mobileNos as keyword and mobile number seperated by commas as value
 $deptSecureKey= "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"; //departsecure key
for encryption of message...
 $encryp_password=sha1(trim($password));
 $templateid= "XXXXXXXXXXXX";//Templateid should be 12 to 19 digit
    //call method and pass value to send single sms, uncomment next line to
use

//sendSingleSMS($username,$encryp_password,$senderid,$message,$mobileno,$dept
SecureKey,$templateid);

    //call method and pass value to send otp sms, uncomment next line to use

//sendOtpSMS($username,$encryp_password,$senderid,$message,$mobileno,$deptSec
ureKey,$templateid);

    //call this method and pass value to send bulk sms, uncomment next line to
use

//sendBulkSMS($username,$encryp_password,$senderid,$message,$mobileNos,$deptS
ecureKey,$templateid);
    //call this method for sending single unicode sms, uncomment next line to
use
```

```
//sendSingleUnicode($username,$encryp_password,$senderid,$messageUnicode,$mob
ileno,$deptSecureKey,$templateid);

    //call this method for sending single unicode otp sms, uncomment next
line to use

//sendUnicodeOtpSMS($username,$encryp_password,$senderid,$messageUnicode,$mob
ileno,$deptSecureKey,$templateid);

    //call this method to send bulk unicode sms, uncomment next line to use

//sendBulkUnicode($username,$encryp_password,$senderid,$messageUnicode,$mobil
eNos,$deptSecureKey,$templateid);

    //function to send sms using by making http connection
     function post_to_url($url, $data) {
     $fields = '';
     foreach($data as $key => $value) {
     $fields .= $key . '=' . $value . '&';
     }
     rtrim($fields, '&');
     $post = curl_init();
     curl_setopt($post,CURLOPT_SSL_VERIFYPEER,false);
     curl_setopt($post, CURLOPT_URL, $url);
     curl_setopt($post, CURLOPT_POST, count($data));
     curl_setopt($post, CURLOPT_POSTFIELDS, $fields);
     curl_setopt($post, CURLOPT_RETURNTRANSFER, 1);
     $result = curl_exec($post); //result from mobile seva server
     echo $result; //output from server displayed
     curl_close($post);
     }

    //function to send unicode sms by making http connection
     function post_to_url_unicode($url, $data) {
     $fields = '';
     foreach($data as $key => $value) {
     $fields .= $key . '=' . urlencode($value) . '&';
     }
     rtrim($fields, '&');

     $post = curl_init();
     curl_setopt($post,CURLOPT_SSL_VERIFYPEER, false);
     curl_setopt($post, CURLOPT_URL, $url);
     curl_setopt($post, CURLOPT_POST, count($data));
     curl_setopt($post, CURLOPT_POSTFIELDS, $fields);
     curl_setopt($post, CURLOPT_HTTPHEADER, array("Content-
Type:application/x-www-form-urlencoded"));
     curl_setopt($post, CURLOPT_HTTPHEADER, array("Content-length:"
    . strlen($fields) ));
     curl_setopt($post, CURLOPT_HTTPHEADER, array("User-Agent:Mozilla/4.0
(compatible; MSIE 5.0; Windows 98; DigExt)"));
     curl_setopt($post, CURLOPT_RETURNTRANSFER, 1);
     echo $result = curl_exec($post); //result from mobile seva server
     curl_close($post);
     }
```

```
//function to convert unicode text in UTF-8 format
function string_to_finalmessage($message){
$finalmessage="";
$sss = "";
for($i=0;$i<mb_strlen($message,"UTF-8");$i++) {
$sss=mb_substr($message,$i,1,"utf-8");
$a=0;
$abc="&#".ordutf8($sss,$a).";";
$finalmessage.=$abc;
}
return $finalmessage;
}

//function to convet utf8 to html entity
function ordutf8($string, &$offset){
$code=ord(substr($string, $offset,1));
if ($code >= 128)
{ //otherwise 0xxxxxxx
if ($code < 224) $bytesnumber = 2;//110xxxxx
else if ($code < 240) $bytesnumber = 3; //1110xxxx
else if ($code < 248) $bytesnumber = 4; //11110xxx
$codetemp = $code - 192 - ($bytesnumber > 2 ? 32 : 0) -
($bytesnumber > 3 ? 16 : 0);
for ($i = 2; $i <= $bytesnumber; $i++) {
$offset ++;
$code2 = ord(substr($string, $offset, 1)) - 128;//10xxxxxx
$codetemp = $codetemp*64 + $code2;
}
$code = $codetemp;

}
return $code;
}

//Function to send single sms
function
sendSingleSMS($username,$encryp_password,$senderid,$message,$mobileno,$deptSe
cureKey,$templateid){

$key=hash('sha512',trim($username).trim($senderid).trim($message).trim($deptS
ecureKey));

$data = array(
"username" => trim($username),
"password" => trim($encryp_password),
"senderid" => trim($senderid),
"content" => trim($message),
"smsservicetype" =>"singlemsg",
"mobileno" =>trim($mobileno),
"key" => trim($key),
"templateid" => trim($templateid)



);
post_to_url("https://msdgweb.mgov.gov.in/esms/sendsmsrequestDLT",$data);
//calling post_to_url to send sms
}
```

```
	//Function to send otp sms
	function
sendOtpSMS($username,$encryp_password,$senderid,$message,$mobileno,$deptSecur
eKey,$templateid){

$key=hash('sha512',trim($username).trim($senderid).trim($message).trim($deptS
ecureKey));

	$data = array(
	"username" => trim($username),
	"password" => trim($encryp_password),
	"senderid" => trim($senderid),
	"content" => trim($message),
	"smsservicetype" =>"otpmsg",
	"mobileno" =>trim($mobileno),
	"key" => trim($key),
	"templateid" => trim($templateid)

	);
	post_to_url("https://msdgweb.mgov.gov.in/esms/sendsmsrequestDLT",$data);
//calling post_to_url to send otp sms
	}


	//function to send bulk sms
	function
sendBulkSMS($username,$encryp_password,$senderid,$message,$mobileNos,$deptSec
ureKey,$templateid){
	$key=hash('sha512',
trim($username).trim($senderid).trim($message).trim($deptSecureKey));

	$data = array(
	"username" => trim($username),
	"password" => trim($encryp_password),
	"senderid" => trim($senderid),
	"content" => trim($message),
	"smsservicetype" =>"bulkmsg",
	"bulkmobno" =>trim($mobileNos),
	"key" => trim($key),
	"templateid" => trim($templateid)

	);
	post_to_url("https://msdgweb.mgov.gov.in/esms/sendsmsrequestDLT",$data);
//calling post_to_url to send bulk sms
	}

	//function to send single unicode sms
	function
sendSingleUnicode($username,$encryp_password,$senderid,$messageUnicode,$mobil
eno,$deptSecureKey,$templateid){
		$finalmessage=string_to_finalmessage(trim($messageUnicode));

$key=hash('sha512',trim($username).trim($senderid).trim($finalmessage).trim($
deptSecureKey));

	$data = array(
	"username" => trim($username),
	"password" => trim($encryp_password),
	"senderid" => trim($senderid),
```

```
"content" => trim($finalmessage),
"smsservicetype" =>"unicodemsg",
```

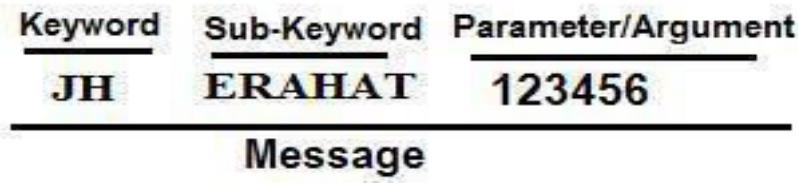
```
        "mobileno" =>trim($mobileno),
        "key" => trim($key),
        "templateid" => trim($templateid)

        );

post_to_url_unicode("https://msdgweb.mgov.gov.in/esms/sendsmsrequestDLT",$data);
//calling post_to_url_unicode to send single unicode sms
        }

        //function to send bulk unicode sms
        function
sendBulkUnicode($username,$encryp_password,$senderid,$messageUnicode,$mobileN
os,$deptSecureKey,$templateid){
        $finalmessage=string_to_finalmessage(trim($messageUnicode));

$key=hash('sha512',trim($username).trim($senderid).trim($finalmessage).trim($
deptSecureKey));

        $data = array(
        "username" => trim($username),
        "password" => trim($encryp_password),
        "senderid" => trim($senderid),
        "content" => trim($finalmessage),
        "smsservicetype" =>"unicodemsg",
        "bulkmobno" =>trim($mobileNos),
        "key" => trim($key),
        "templateid" => trim($templateid)

        );

post_to_url_unicode("https://msdgweb.mgov.gov.in/esms/sendsmsrequestDLT",$data);
//calling post_to_url_unicode to send bulk unicode sms
    }

    //function to send single unicode otp sms
        function
sendUnicodeOtpSMS($username,$encryp_password,$senderid,$messageUnicode,$mobil
eno,$deptSecureKey,$templateid){
            $finalmessage=string_to_finalmessage(trim($messageUnicode));

$key=hash('sha512',trim($username).trim($senderid).trim($finalmessage).trim($
deptSecureKey));

        $data = array(
        "username" => trim($username),
        "password" => trim($encryp_password),
        "senderid" => trim($senderid),
        "content" => trim($finalmessage),
        "smsservicetype" =>"unicodeotpmsg",
        "mobileno" =>trim($mobileno),
        "key" => trim($key),
        "templateid" => trim($templateid)

        );

post_to_url_unicode("https://msdgweb.mgov.gov.in/esms/sendsmsrequestDLT",$data);
```

```
//calling post_to_url_unicode to send single unicode sms
    }
?>
```

Overview

Shortcode 51969 has been allocated for SMS services by the Department of Telecom, Government of Inda to Department of IT, Government of India for providing Government Services on SMS. This shortcode is the single point of access for all the pull based sms services.

Following is the format of SMS Pull request:



SMS PULL Account Creation

For SMS Pull service, Departments need to provide keyword followed by subkeyword which identifies the service. Citizen who wants to avail this service will send SMS to 166 / 51969 with message as Keyword Subkeyword parameter.

Department needs to provide following details for SMS Pull service:

Parameter	Description
Department Name	*Name of the Department*
UserName	For login to MSDG Portal (Use alphabets and numbers only. in of 6 & Max of 15 chars)
Password	For login password (Use alphabets and numbers only. Min of 6 & max of 10 chars)
Contact Person Name	Details of the Contact Person
Address	Address of the Department
Mobile Number	*For verification and alert messages*
Alternate Mobile Numb er	*For verification and alert messages*
Email-ID	*For verification and alert mails*
Project Details	Details of the project or services
Keyword	If department is from some State then keyword should be State code (MH for Maharashtra, UP for Uttar Pradesh, etc). If department is from Central, then keyword can be suggested by the department.It is recommended that keywords should not be of more than 4 characters.
Sub-Keyword	Sub-Keywords: Names of Services of the department. E.g."RATIONC" for ration card application tracking, "BIRTHCR" for birth certificate, etc

URL	Public URL of the interface of Department service in HTTP.Department must clearly specify whether the provided HTTP interface is GET or POST.
IP	Public IP of the server where the department service is hosted. This is required by our data center for white-listing this particular IP. For security reasons, MSDG server makes call to only those servers,whose IPs are white-listed in our data center.
Ports	We assume that department service is running on 80 or 443 port(s).Provide the ports if it is other than 80 or 443.

SMS PULL Parameter Definitions:

SMS gateway of MSDG receives the following information from the Mobile Network Operator (Telcos):

- Mobile Number (current supported format is: 9324692411)
- Time Stamp (in the format "yyyy-mm-dd hh:mm:ss")
- Operator Name (currently not being provided by the Mobile Network Operator)
- Area Code (currently not being provided by the Mobile Network Operator)
- Message (complete 160 characters)

The above details are forwarded to the department as it is, in the format as provided in the example below (the department URL is HTTP GET). Currently Operator Name and the Area Code will be sent to the department as blank.

The interface provided by the department must have following parameters:

Parameter	Description
Mobile Number	Mobile number of requester
TimeStamp	Time Stamp (in the format "yyyy-mm-dd (h:mm:ss") of the request
OperatorName	Name of the service provider of the requester (currently not being provided by the Mobile Network Operator)
AreaCode	Area code of the requester (currently not being provided by the Mobile Network Operator)
Message	The complete message received by SMS gateway. (KEYWORD+ SUBKEYWORD+ message)

Example of PULL Request:

http://department.gov.in/sms?mobileNumber=987654321&timeStamp=2012-02-23 13:30:20&operatorName=&areaCode=&message=KEYWORD SUBKEYWORD 1234567890123

For secure Pull Service:

Parameter	Description
Mobile Number	*Mobile number of requester*
TimeStamp	Time Stamp (in the format"yyyy-mm-dd hh:mm:ss") of the request
OperatorName	Name of the service provider of the requester (currently not being provided by the Mobile Network Operator)
AreaCode	Area code of the requester (currently not being provided by the Mobile Network Operator)
Message	The complete message received by SMS gateway. (KEYWORD + SUBKEYWORD + message)
HashValue	Hash value is generated by using SHA512 hash function of SHA2 family with the combination of message, mobile number, timestamp and secure key(which is generated at the time of pull registration)

Example of Secure PULL Request:

http://department.gov.in/sms?mobileNumber=<mobileNumber>&timeStamp=<timeStamp>&operatorName=<operatorName>&areaCode=<areaCode>&hash=<hashvalue>&message=<KEYWORD SUBKEYWORD message>

How to Choose Keywords and Sub-Keywords?

To make shortcode 51969 services easier to use, a citizen should not have to remember complicated keywords and sub-keywords for a service. A good shortcode service thus has a very flat hierarchy and should be simple to explain in the length of a single text message.
A suggested configuration has been described below.

Keywords: Names of States.

Sub-Keywords: Names of Services and parameters/arguments.

Responses: Usually less than a single text message.

Example:

When the citizen sends an SMS "GOA RATIONC XXXX" to the short-code 51969, the first word represents the keyword for the states, the second word RATIONC represents for keyword for the Ration Card service and the third word represent the application number.

Recommendation:

It is also recommended that every keyword has a configured HELP sub-keyword for service discovery. In case of an invalid SMS being sent, an instruction to use the HELP discovery service should be sent back.

How to Frame Messages

These following rules of thumb are useful when framing messages to send in response to citizen queries:

Messages should be short. When possible, fit them within the length of one SMS message.

Do not use SMS lingo. While popular in personal messaging, studies have shown that citizens do not expect service messages to be in SMS lingo. Use professional language, and meaningful phrases.

Use helpful error messages. Because composing SMS is a time-consuming process, guide the citizen whenever possible to complete his query. The SMS application should, whenever possible interpret citizen's queries and give a response, regardless of his particular query format.

Chapter 3 : IVRS Pull Integration

Overview

Shortcode 166 has been allocated for all MSDG services by the Department of Telecom, Government of India to Department of Electronics and IT, Government of India for providing Government Services on various mobile based channels. This shortcode will be the single point of access for all the MSDG services.

This shortcode will be used for MSDG IVRS services. Currently we are in the process of integrating with various telecom operators for 166. Currently our IVRS system is running and operational on 022-26209367

The departments which want to put their services on IVRS, a dial plan will be created and will be added to existing IVRS menu. If a department is from some State, then its services will be under the concerned State menu. If it is central government department it will come under central government services menu.

Departments need to publish an interface which will be called when citizen calls up IVRS number for a department service.

IVRS Parameter Definitions:

MSDG receives the following information from the Mobile Network Operator (Telco) for an IVRS request:

- Mobile Number / phone number
- Time Stamp (in the format "yyyy-mm-dd hh:mm:ss")
- Operator Name (currently not being provided by the Mobile Network Operator)
- Area Code (currently not being provided by the Mobile Network Operator)
- Message (numeric values only as citizen can enter only numeric inputs)

The above details are forwarded to the department as it is, in the format as provided in the example below (the department URL is HTTP GET). Currently Operator Name and the Area Code will be sent to the department as blank.

Citizen can be call to our IVRS services for accessing the services of the departments. Particulars services should be selected from the IVRS menu. Citizen then Enter the valid parameter of the selected service. The IVRS System forward these information like mobile number and valid parameter to the department through published interface. The Departments should return response in the form of text to IVRS system .IVRS system simply converts this Text response into speech response which is audible to citizen.

The interface provided by the department must have following parameters

Parameter	Description
Mobile Number	Mobile number of requester
TimeStamp	Time Stamp (in the format "yyyy-mm-dd hh:mm:ss") of the request
OperatorName	Name of the service provider of the requester (currently not being provided by the Mobile Network Operator)
AreaCode	Area code of the requester (currently not being provided by the Mobile Network Operator)
Message	Keyword + " " + SubKeyword + " " + The message received by MSDG IVRS servers.

Note: We have kept the format of the interface same as that of SMS, so that department can make their services available on multiple channels (SMS, IVRS, and USSD) through same interface. So if a service is available on SMS, it can be made available on IVRS and vice versa.

Example of IVRS Request:

keyword=<PutKeywordhere>&subkey=<PutSubKeywordhere>&mobilno=9876543210&message=<PutKeywordhere><PutSubKeywordhere>1234567890123&operatorname=ss&areacode=SS " http://msdgweb.mgov.gov.in/esms/smspullrequest

Chapter 4 : IVRS Push Integration

Overview

IVRS is an example of a computer-telephone integration (CTI). The most common way for a phone to communicate with a computer is through the tones generated by each key on the telephone keypad. These are known as dual-tone multi-frequency (DTMF) signals. A computer needs special hardware called a telephony board or telephony card to understand the DTMF signals produced by a phone.A simple IVR system only requires a computer hooked up to a phone line through a telephony board and some IVR software. The IVR software allows pre-recording of greetings and menu options that a caller can select using his telephone keypad. More advanced IVR systems include speech-recognition software that allows a caller to communicate with a computer using simple voice commands. Speech recognition software has become sophisticated enough to understand names and long strings of numbers.

In the context of mobile governance, the IVRS application is intended to serve the C2G and G2C services within the e-governance domain. Through IVRS based services, status enquiries for a large number of services can be automated and the requisite information provided to the service seekers without causing undue overheads on the e-governance infrastructure. API used by departments for generating calls using HTTPS URL interface. The API supports single as well as multiple calls with number separated by comma(,).The url used is

https:// services.mgov.gov.in/PushCallAPI/MakePushCall.

IVRS Account Creation

For IVRS integration, Departments need to provide following details:

Parameter	Description
Department Name	*Name of the Department*
UserName	For login to MSDG Portal (Use alphabets and numbers only. in of 6 & Max of 15 chars)
Password	For login password (Use alphabets and numbers only. Min of 6 & max of 10 chars)
Contact Person Name	Details of the Contact Person
Address	Address of the Department
Mobile Number	*For verification and alert messages*
Alternate Mobile Number	*For verification and alert messages*
Email-ID	*For verification and alert mails*
Project Details	Details of the project or services
Keyword	If department is from some State then keyword

	should be Statecode (MH for Maharashtra, UP for Uttar Pradesh etc).If department is from Central, then keyword can be suggested by the department.It is recommended that keywords should not be of more than 4 characters.
Sub-Keyword	Sub-Keywords: Names of Services of the department. E.g."RATIONC" for ration card application tracking, "BIRTHCR" for birth certificate etc
URL	Public URL of the interface of their service in HTTP. Department must clearly specify whether the provided HTTP interface is GET or POST.
IP	Public IP of the server where the department service is hosted. This is required by our data center for white-listing this particular IP. For security reasons, MSDG server makes call to only those servers, whose IPs are white-listed in our data center.
Ports	We assume that department service is running on 80 or 443 port(s).Provide the ports if it is other than 80 or 443.

Term and Defination

User Name: The Registered User Name on Services portal.
PassWord: Login Password of Services Portal.
Voice code: The voice file that already uploaded on services portal and would be play in the generated calls.

Java Example:

MakePushCalls.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.StringWriter;
import java.io.UnsupportedEncodingException;
import java.net.URL;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.List;

import javax.net.ssl.SSLContext;

import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.scheme.Scheme;import
org.apache.http.conn.ssl.SSLContexts; import
org.apache.http.conn.ssl.SSLSocketFactory;
```

```
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;

public class MakePushCall {



    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String userName="";//Your username of services portal
        String password="";//Your Password
        String mobileNumber="";//If more than 1 then seprated by comma(,)
        String voiceCode="";//as uploaded on website( give only nuber of
voicecode)
        makePushCall(userName,password,mobileNumber,voiceCode);

    }



  private static void makePushCall( String userName,String password,String
mobileNumbers,String voiceCode){

        SSLSocketFactory sf=null;
        SSLContext context=null;
        try {
            context=SSLContext.getInstance("TLSv1.2");
            context.init(null, null, null);
            sf=new SSLSocketFactory(context,
SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER);
            Scheme scheme=new Scheme("https",443,sf);
            @SuppressWarnings("resource")
            HttpClient client=new DefaultHttpClient();

client.getConnectionManager().getSchemeRegistry().register(scheme);
            HttpPost post=new
HttpPost("https://services.mgov.gov.in/PushCallAPI/MakePushCall");
            List<NameValuePair> nameValuePairs=new
ArrayList<NameValuePair>(1);
            nameValuePairs.add(new BasicNameValuePair("username", userName));
            nameValuePairs.add(new BasicNameValuePair("password", password));
            nameValuePairs.add(new BasicNameValuePair("MobileNumbers",
mobileNumbers));
            nameValuePairs.add(new BasicNameValuePair("voiceCode",
voiceCode));
            post.setEntity(new UrlEncodedFormEntity(nameValuePairs));
            HttpResponse response=client.execute(post);
            BufferedReader bf=new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
            String line="";
            while((line=bf.readLine())!=null){
                System.out.println("response==>"+line);
            }

        } catch (NoSuchAlgorithmException e) {
```

```
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (KeyManagementException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ClientProtocolException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

}
```

MakePushCalls.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Web;
using System.Net;
using System.IO;
using System.Security;
using System.Security.Cryptography.X509Certificates;


namespace ConsoleApplication3

    class Program
    {
        static void Main(string[] args)
        {


            String username = "" String
            password = ""; String
            MobileNumbers = ""; String
            voiceCode = "";


            String postData = "username=" + username + "&password=" +
password + "&MobileNo=" + MobileNumbers + "&CallID"+CallID +"&voiceCode=" +
voiceCode;//data to post
            String url =
"https://services.mgov.gov.in/PushCallAPI/MakePushCall";//url

            System.Net.ServicePointManager.SecurityProtocol =
SecurityProtocolType.Tls12 | SecurityProtocolType.Tls11 |
SecurityProtocolType.Tls;
            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
```

```
            request.KeepAlive = false;
            request.ProtocolVersion = HttpVersion.Version10;
            request.Method = "POST";
            System.Net.ServicePointManager.CertificatePolicy = new
MyPolicy();

            byte[] array = Encoding.ASCII.GetBytes(postData);

            request.ContentType = "application/x-www-form-urlencoded";
            request.ContentLength = array.Length;
            Stream requestStream = request.GetRequestStream();


            requestStream.Write(array, 0, array.Length);
            requestStream.Close();

            HttpWebResponse response =
(HttpWebResponse)request.GetResponse();
            Console.WriteLine(new
StreamReader(response.GetResponseStream()).ReadLine());
            Console.WriteLine(response.StatusCode);
            Console.ReadLine();

        }
        class MyPolicy : ICertificatePolicy
        {
            public bool CheckValidationResult(ServicePoint srvPoint,
X509Certificate certificate, WebRequest request, int certificateProblem)
            {
                return true;
            }
        }
    }
}


PHP Code

<?Php

        function post_to_url() {
        $fields = '';
        $url="https://services.mgov.gov.in/PushCallAPI/MakePushCall";

        $data = array(
           "username" => "", //username  of department
           "password" => "", // password of department
           "MobileNumbers" =>"",     // for ex 9988776644
           "voiceCode" =>"", // voice code of file

        foreach($data as $key => $value) {
           $fields .= $key . '=' . $value . '&';
        }
        rtrim($fields, '&');

        $post =curl_init();
```

```
        curl_setopt($post, CURLOPT_URL, $url);
        curl_setopt($post, CURLOPT_SSL_VERIFYPEER, false);
        curl_setopt($post, CURLOPT_POST, count($data));
        curl_setopt($post, CURLOPT_POSTFIELDS, $fields);
        curl_setopt($post, CURLOPT_RETURNTRANSFER, 1);
        $result =curl_exec($post);
        echo"$result";
        if(curl_errno($post))
        {
            echo 'error:' . curl_error($post);
        }
         curl_close($post);

}
        post_to_url();

?>
```

API Response Codes

Response Code	Meaning
401	Credentials Error, may be invalid username or password
402	Call submitted successfully
403	Credits not available

Chapter 5 : Unstructured Supplementary Services Data (USSD)

USSD is a session based service unlike SMS which is a store and forward service. USSD services are provided with two different service features:

★USSN (Unstructured Supplementary Services Notify)

This type of service is also known as flash push notification service. USSD will send an acknowledgement to the application once the flash is submitted to the handset. These acknowledgements are flash based.This type of service is also known as flash push notification service. Under this feature departments are allowed to send messages to the subscribers/end users. These are simple FLASH to the handset. Subscribers cannot respond to these messages. These are non-menu based and are used just for a notification or an info

★USSR (Unstructured Supplementary Services Response)

This type of service is also known as flash pull notification service.Under this feature, departments have the facility to send menu to the subscribers in place of a simple flash. The department has to imitate the messages as a Push to the handset with defined menu and actions. After receiving the menu, subscribers/end-users can send the response over as a second session. Thus under USSR, a department has to have a menu and functional options for each session. These are more interactive and application related. The USSR service will be available on *166#.

Note: Currently we are providing USSR to the citizens.

USSD Requirement from Department

1. Public IP of the USSD service Server.
2. Port number of USSD Server.
3. Public URL of the service which is deployed .We need a URL which accepts parameters as HTTP get/post entity as follows:

Parameter	Description
mobileNumber	Mobile number of requester
choice	It will be the user input provided to the *166#
level	Level is maintained for tracking level of menu. Please refer to below example for more clarity.

Note: Please provide a single string (your menu level and menu with "$%^" in between) in HTTP response .

For Example:

If level 1 menu is "Welcome to Mobile Seva" then http response string should be "1"+"$%^"+"Welcome to Mobile Seva" ("1$%^Welcome to Mobile Seva").

Note: When department provide final menu in response it should be like "END"+"$%^"+final menu.

We also need the Server IP and PORT on which URL is running on for white listing at our end.

Java Example for USSD

```
package in.cdac.ussdtest;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**

* Servlet implementation class USSDServlet */

public class USSDServlet extends HttpServlet {

private static final long serialVersionUID = 1L;



/**

* @see HttpServlet#HttpServlet() */



public USSDServlet() { super();

// TODO Auto-generated constructor stub

}

/**

* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response) */

protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException{

}

/**

* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response) */

protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws

ServletException, IOException {

//  TODO Auto-generated method stub Select 2 for XXXX Service";

//  TODO Auto-generated method stub
```

```
PrintWriter out=response.getWriter();

String mobileNo=request.getParameter("mobileNumber"); String
choice=request.getParameter("choice");

String level=request.getParameter("level");

String Menu1="Welcome to XXXX Department Services.\n Select 1 for XXXX
Service.\n String Menu2="Please Enter Your Application Number";

String FinalMenu="Thank you for Applying for XXXX Service.We will get back
you soon";

//System.out.println("Mobile Number ::::"+ mobileNo);
if(level.equals("start")){

out.println("1$%^"+Menu1);

///do some work here
}

else if(level.equals("1")&&choice.equals("1"))
{

System.out.println("Application XXXXX");


out.println("2$%^"+Menu2);
///do some work here

}

else if (level.equals("1")&&choice.equals("2"))

{

System.out.println("Application XXXXX"); out.println("2$%^"+Menu2);

///do some work here
}

else { out.println("END$%^"+FinalMenu);

///do some work here

}
}

}
```