Mobile Seva

MSDP Services
Integration Document

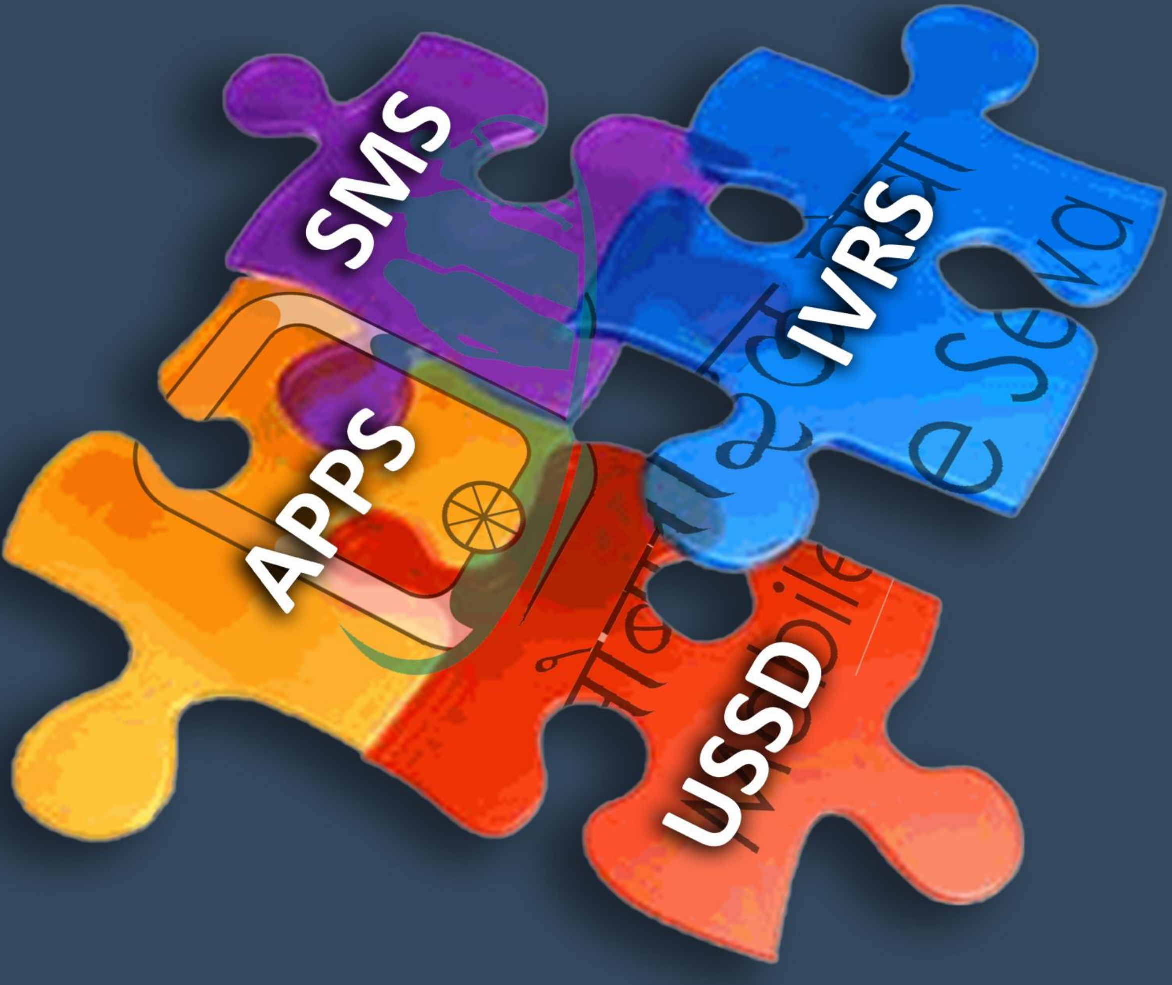


Table of Contents

Push SMS Integration

1.1 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.

1.2 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 Sender ID 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.

1.3 Using the HTTP URL for sending messages:

The end point of the service is http://msdgweb.mgov.gov.in/esms/sendsmsrequest. 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.

1.4 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)

1.5 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 attached to the username*
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*
starttime	*For schedule message in WSDL service.* *Mention this for scheduling messages. Messages will be sent at the set time* *TIME FORMAT YYYYMMDD hh:mm:ss and time GMT ie IST - 05:30 hours for example if you want to schedule sms for Jan 1 2009 08:00:00PM you should enter start time as 2009-01-01 14:30:00*
endtime	*Leave this field blank, not relevant currently*
messages	*This is for using sendpairedsms method in webservice.* *<message>* *<text>Test Message</text>* *<numbers>919000000000</numbers>* *</message>*

1.6 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

1.7 Java Example

SMSHttpPostClient.java

```
package in.gov.mgov.msdg.sms;

import java.io.*;
import java.net.*;

public class SMSHttpPostClient {
        static String username = "username";
        static String password = "password";
        static String senderid = "sender_id";
        static String message = "Test SMS from MSDG, Sorry for inconvenience!";
        static String mobileNo = "09324596412";
        static String mobileNos = "09324596412,09324596412";
        // StartTime Format: YYYYMMDD hh:mm:ss
        static String scheduledTime = "20110701 02:27:00";
        static HttpURLConnection connection = null;
```

```
    public static void main(String[] args) {

        try {
            URL url = new URL("http://msdgweb.mgov.gov.in/esms/sendsmsrequest");
            connection = (HttpURLConnection) url.openConnection();
            connection.setDoInput(true);
            connection.setDoOutput(true);
            connection.setRequestMethod("POST");
            connection.setFollowRedirects(true);
            // connection = sendSingleSMS(username, password, senderid,
            // mobileNo, message);
            // connection = sendBulkSMS(username, password, senderid, mobileNos,
            // message);
            connection = sendScheduledSMS(username, password, senderid,
                        mobileNos, message, scheduledTime);

            System.out.println("Resp Code:" + connection.getResponseCode());
            System.out.println("Resp Message:"
                        + connection.getResponseMessage());

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


// Method for sending single SMS.
public static HttpURLConnection sendSingleSMS(String username,
            String password, String senderId,
            String mobileNo, String message) {
    try {
        String smsservicetype = "singlemsg"; // For single message.
            String query = "username=" + URLEncoder.~~encode~~(username)
                + "&password=" + URLEncoder.~~encode~~(password)
                + "&smsservicetype=" + URLEncoder.~~encode~~(smsservicetype)
                + "&content=" + URLEncoder.~~encode~~(message) + "&mobileno="
                + URLEncoder.~~encode~~(mobileNo) + "&senderid="
                + URLEncoder.~~encode~~(senderId);

        connection.setRequestProperty("Content-length", String
            .valueOf(query.length()));
        connection.setRequestProperty("Content-Type",
            "application/x-www-form-urlencoded");
        connection.setRequestProperty("User-Agent",
            "Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)");

        // open up the output stream of the connection
            DataOutputStream output = new DataOutputStream(connection
                        .getOutputStream());

        // write out the data
```

```
                int queryLength = query.length();
                output.writeBytes(query);
                // output.close();

                // get ready to read the response from the cgi script
                DataInputStream input = new DataInputStream(connection
                                        .getInputStream());

                // read in each character until end-of-stream is detected
                for (int c = input.read(); c != -1; c = input.read())
                        System.out.print((char) c);
                input.close();
        } catch (Exception e) {
                System.out.println("Something bad just happened.");
                System.out.println(e);
                e.printStackTrace();
        }

        return connection;
}


// method for sending bulk SMS
public static HttpURLConnection sendBulkSMS(String username,
                String password, String senderId, String mobileNos, String message) {
        try {
                String smsservicetype = "bulkmsg"; // For bulk msg
                String query = "username=" + URLEncoder.~~encode~~(username)
                        + "&password=" + URLEncoder.~~encode~~(password)
                        + "&smsservicetype=" + URLEncoder.~~encode~~(smsservicetype)
                        + "&content=" + URLEncoder.~~encode~~(message)
                        + "&bulkmobno=" + URLEncoder.encode(mobileNos, "UTF-8")
                        + "&senderid=" + URLEncoder.~~encode~~(senderid);

                connection.setRequestProperty("Content-length", String
                        .valueOf(query.length()));
                connection.setRequestProperty("Content-Type",
                        "application/x-www-form-urlencoded");
                connection.setRequestProperty("User-Agent",
                        "Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)");

                // open up the output stream of the connection
                DataOutputStream output = new DataOutputStream(connection
                        .getOutputStream());

                // write out the data
                int queryLength = query.length();
                output.writeBytes(query);
                // output.close();

                System.out.println("Resp Code:" + connection.getResponseCode());
                System.out.println("Resp Message:" + connection.getResponseMessage());

                // get ready to read the response from the cgi script
                DataInputStream input = new DataInputStream(connection
                        .getInputStream());

                // read in each character until end-of-stream is detected
                for (int c = input.read(); c != -1; c = input.read())
                        System.out.print((char) c);
                input.close();
                } catch (Exception e) {
                        System.out.println("Something bad just happened.");
                        System.out.println(e);
                        e.printStackTrace();
                }
                return connection;
        }

// method for sending the scheduled SMS
public static HttpURLConnection sendScheduledSMS(String username, String password,
        String senderId, String mobileNos, String message, String scheduledTime) {

        try {
                String smsservicetype = "schmsg"; // For Scheduled message.

                String query = "username=" + URLEncoder.~~encode~~(username)
                        + "&password=" + URLEncoder.~~encode~~(password)
                        + "&smsservicetype=" + URLEncoder.~~encode~~(smsservicetype)
                        + "&content=" + URLEncoder.~~encode~~(message)
                        + "&bulkmobno=" + URLEncoder.encode(mobileNos, "UTF-8")
                        + "&senderid=" + URLEncoder.~~encode~~(senderid) + "&time="
                        + URLEncoder.encode(scheduledTime, "UTF-8");

                connection.setRequestProperty("Content-length", String
                        .valueOf(query.length()));
                connection.setRequestProperty("Content-Type",
                        "application/x-www-form-urlencoded");
                connection.setRequestProperty("User-Agent",
                        "Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)");

                // open up the output stream of the connection
                DataOutputStream output = new DataOutputStream(connection
                                        .getOutputStream());

                // write out the data
                int queryLength = query.length();
                output.writeBytes(query);
                // output.close();

                System.out.println("Resp Code:" + connection.getResponseCode());
                System.out.println("Resp Message:"
                                        + connection.getResponseMessage());

                // get ready to read the response from the cgi script
                DataInputStream input = new DataInputStream(connection
                        .getInputStream());

                // read in each character until end-of-stream is detected
                for (int c = input.read(); c != -1; c = input.read())
```

```
                        System.out.print((char) c);
                        input.close();
                } catch (Exception e) {
                        System.out.println("Something bad just happened.");
                        System.out.println(e);
                        e.printStackTrace();
                }
                return connection;
        }
}
```

1.8 .NET (C#) Example

SMSHttpPostClient.cs

```
using System;
using System.Text;
using System.Net;
using System.Web;
using System.IO;

namespace esms_client
{
   public class SMSHttpPostClient
   {
      static String username = "username";
      static String password = "password";
      static String senderid = "senderid";
      static String message = "message";
      static String mobileNo = "9856XXXXX";
      static String mobileNos = "9856XXXXX, 9856XXXXX ";
      static String scheduledTime = "20110819 13:26:00";
      static HttpWebRequest request;
      static Stream dataStream;
      public static void Main(String[] args)
      {
         request = (HttpWebRequest)WebRequest.Create("http://msdgweb.mgov.gov.in/esms/sendsmsrequest");
         request.ProtocolVersion = HttpVersion.Version10;
         //((HttpWebRequest)request).UserAgent = ".NET Framework Example Client";
         ((HttpWebRequest)request).UserAgent="Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)";
         request.Method = "POST";
         Console.WriteLine("Before Calling Method");
         sendSingleSMS(username, password, senderid, mobileNo, message);
        sendBulkSMS(username, password, senderid, mobileNos, message);
            sendScheduledSMS(username, password, senderid, mobileNos, message, scheduledTime);

      }

// Method for sending single SMS.

      public static void sendSingleSMS(String username, String password, String senderid,
         String mobileNo, String message)
      {
         String smsservicetype = "singlemsg"; //For single message.
         String query = "username=" + HttpUtility.UrlEncode(username) +
            "&password=" + HttpUtility.UrlEncode(password) +
            "&smsservicetype=" + HttpUtility.UrlEncode(smsservicetype) +
            "&content=" + HttpUtility.UrlEncode(message) +
            "&mobileno=" + HttpUtility.UrlEncode(mobileNo) +
            "&senderid=" + HttpUtility.UrlEncode(senderid);

         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();
      }
// method for sending bulk SMS
      public static void sendBulkSMS(String username, String password, String senderid,String mobileNos, String
message)
      {
            String smsservicetype = "bulkmsg"; // for bulk msg
         String query = "username=" + HttpUtility.UrlEncode(username) +
            "&password=" + HttpUtility.UrlEncode(password) +
            "&smsservicetype=" + HttpUtility.UrlEncode(smsservicetype) +
            "&content=" + HttpUtility.UrlEncode(message) +
            "&bulkmobno=" + HttpUtility.UrlEncode(mobileNos) +
            "&senderid=" + HttpUtility.UrlEncode(senderid);
            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();
             }
}
```

1.9 PHP Example

SMSHttpPostClient.php

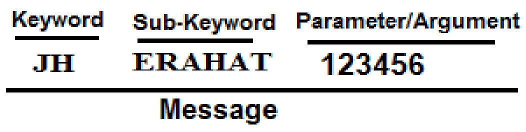
```
<?php

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

   $post = curl_init();

   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);

   echo $result = curl_exec($post);

   curl_close($post);
  }

  $data = array(
   "username" => "username",              // type your assigned username here(for example:
                                        "username" => "CDACMUMBAI")

   "password" => "password",          //type your password

   "senderid" =>"senderid",           //type your senderID

   "smsservicetype" =>"singlemsg",     //*Note*  for single sms enter  "singlemsg" , for bulk
                                            enter "bulkmsg"

   "mobileno" =>"mobileno",           //enter the mobile number

   "bulkmobno" => "bulkmobno",              //enter the mobile numbers separated by commas, in
                                             case of bulk sms otherwise leave it blank

   "content"  => "message"            //type the message.

  );

  post_to_url("http://msdgweb.mgov.gov.in/esms/sendsmsrequest", $data);


?>
```

Pull SMS Integration

2.1 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:



2.2 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.*

2.3 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 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)*

2.4 Example of PULL Request:

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

2.5 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)*

2.6 Example of Secure PULL Request:

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

2.7 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.

IVRS Pull Integration

3.1 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.

3.2 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.

3.3 Example of IVRS Request

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

IVRS Push Integration

4.1 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.

4.2 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 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 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.*

4.3 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.

4.4 Java Example

MakePushCall.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="xyz";//Your username of services portal
		String password="xyz@123";//Your Password of services portal
		String mobileNumber=”XXXXXXXXXX,XXXXXXXXXX”;//If more than 1 then
seprated by comma(,)
		String voiceCode="1";//as uploaded voice file on website
		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("TLS");
			context.init(null, null, null);
			sf=new SSLSocketFactory(context,
~~SSLSocketFactory~~.ALLOW_ALL_HOSTNAME_VERIFIER);
			~~Scheme~~ scheme=new Scheme("https",443,sf);
			HttpClient client=new DefaultHttpClient();

	client.~~getConnectionManager~~().getSchemeRegistry().register(scheme);
			HttpPost post=new
HttpPost("https://164.100.129.131/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();
            }
        }
}
```

4.5 API Response Codes

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

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.

5.1 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.

5.2 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

}
}
}
```