Android oreo send sms programmatically



With so many different ways of communicating (I 100 on you Google Developers), you could find a strange talk about integrating an old-school SMS service into your app. Admittedly, aside from the usual 2-step verification and related services, there are some areas on the planet related services), you could find a strange talk about integrating an old-school SMS service into your app. expensive, so if we want to solve the next billion, most of these new users are in emerging markets so SMS can be a good alternative or at least something to consider. TL;DR: The project is available on GitHub. Note and review gives you the rest. With relatively little effort, you can have your app via SMS as a data source, while providing a nice UX user interface, a user interface for material design (beats simple text). Note: I use project Bus-eta as an example of implementation, you can read more about it in my previous post. I also created a simplified and easy-to-understand project, check available on GitHub.OverviewAsk for SMS permissions (like manifest and running time)Send SMS via the SmsManagerSetting system to BroadcastReceiver to listen and process for incoming SMS.1. Request permission in order to To make our app work with SMS, Start with ads in your AndroidManifestNow our users before M will be asked to accept these permissions when they decide to install the app through PlayStore.But as the launch in Android 6.0 (API level 23), permissions must be requested at the time of execution, so we still have to make our zefir app ready, otherwise a large percentage of users will not be able to use sms, resulting in very frustrating and effortless. Sending an SMS when you don't ask permission, if not, request: It's important not to ask in advance, I personally hate it when I start a new app and I get asked for random permissions, sometimes I don't even use the app, so don't be lazy and just ask when it's necessary. The permission request will be given to the user by the subsequent AlertDialog: Transport ETA system with a request to allow an SMS message during execution. As you can see the message given by the system is pretty strong, if it's just for your users why your app will need to access SMS. I would recommend you show a simple explanation in advance to avoid some of your more paranoid users (like me) to deny and make the whole feature useless. Give the user some context, simply: By showing a simple explanation in advance With minimal effort, our users are presented with simple but important information. Now they why they should allow the app to access such sensitive data. Once the user decides that we have to process the resolution response: Note: Like the startActivityForResult call, SMS PERMISSION CODE is just a permanent variable, it must comply with both asking permission, and processing Sending SMS through our app Now that we have the necessary permissions we will use the default device's sms messenger app to send sms. (2) 1/9. It is considered good practice to let the user know the normal fees apply, our app is simply dolled up to the version of the usual/default process. 12. Listening to incoming SMSNowadays apps rarely have a single-track flow of communication, so in order to listen to incoming SMS we have to customize our custom BroadcastReceiverBroadcast receiver, which listens to incoming sms'sOf of course, you could always create a simple BroadcastReceiver and logic present on your activities, but I prefer to keep this wrapper under the custom SmsBroadcastReceiver and only when transferring. Avoiding ugly code floating around and unnecessary connection. Looking at line 47 of the aforementioned essence you can see that I have implemented some more stringent conditions when the listener is called, it suits my particular case, I only want to discover sms that come from a certain number and have the initial model of the SMS@CARRIS that is transmitted through the designer: So we can ignore any other incoming sms, Now we need to register the ou receiver, in my case I want to listen while the app is open, so I will register in my application class The positioning step assigns listener, will only be called if the conditions are met: You can also register your transmission receiver through the manifest by adding the following line to your AndroidManifest: And adjust all the logic within your own BroadcasReceiver, so you keep all your business logic wrapped inside your own class. In my case I want to have more control and adaptability, but using the manifest is just like registering in an app. My actual project has a controller that handles a lot of basic edge cases that can pop up with these custom implementations but are out of scope of the article. When using BroadcastReceivers, beware of priority settings (but don't abuse it 🖏). Links:Hello! My name is Joaquim Lei, I am currently based in Copenhagen, Denmark. If you liked this hit article that If you know who can extract value, please share, I will look for my opinion and suggestions in the comments, feedback is always welcome. If you want to see more content from me, abutton to follow! Manages SMS operations such as data, text and PDU SMS messages. Get this object by calling the static getDefault method. To create a copy of SmsManager associated with a specific subscription ID, call (int). This is commonly used for devices that support multiple active subscriptions at the same time. For information on how to behave like the default SMS app on Android 4.4 (API level 19) and above, see Telephony Telephony. SmsManager. Financial SmsCallback call back to provide asynchronous sms messages for the financial application. Line create AppSpecificSmsToken (PendingIntent intention) Create a single use application specific incoming SMS reguest to call the package. The line createAppSpecificSthSTokenWithPackageInfo (String prefixes, pendingIntent intentions) Create one application using a specific incoming SMS request to call the package. Divide the text message into several fragments, no larger than the maximum SMS size. Void downloadMultimediaMessage (Context Context, String LocationUrl, Uri contentUri, Bundle configOverrides, PendingIntent downloadIntent) Download mmS message from the carrier at this location URL Note: This method will never trigger SMS dialogue. GetCarrierConfigValues kit () Get operator-dependent mmS configuration values. static SmsManager getDefault () Get SmsManager associated with the default subscription ID. Static SmsManager getSmanagerForSubscriptionID () Get the default subscription ID. Static SmsManager getSmanager ID. emptiness getSmsMessagesForFinancialApp (Bundle params, Artist, SmsManager.FinancialScallback callback) Get SMS messages to call. GetSmscAddress () Receives SMSC from (U)SIM. getSubscriptionId () Get a related subscription id. void injectSpsPdu (byte, line format, PendingIntent) will ingest SMS PDU into the android app platform. invalid sendDataMessage (String destinationAddress, String scAddress, short destinationPort, data byte, PendingIntent deliveryIntent) Send SMS based on data to a specific application port. invalid sendMultimediaMessage (Context, Uri contentUri, String locationUrl, Bundle configOverrides, PendingIntents, ArrayList'lt;PendingIntents, ArrayList'lt;PendingIntents, ArrayList'lt;PendingIntents; deliveryIntents) String scAddress, ArrayList'lt;String)It; parts, ArrayList'lt;PendingIntents, ArrayList'lt;PendingIntents; deliveryIntents) Send a multi-part text based on SMS. Void sendMultipartTextMessage (StringAddress, Line scAddress, Parts List, List of Parts, List of qlt;PendingIntents, deliveryIntents list, long messageId) Send a multi-part text based on SMS. Void sendMultipartTextMessage (StringAddress, Parts List, List of qlt;PendingIntents, deliveryIntents, deliveryIntents list, long messageId) Send a multi-part text based on SMS. Void sendMultipartTextMessage (StringAddress, Parts List, List of qlt;PendingIntents, deliveryIntents, delivery String scAddress, Parts List, List, List, List, List, List, List, RendingIntents, DeliveryIntents, PackageName Line, Line Аналогичный метод, как #sendMultipartTextMessage (String, ArrayList, ArrayList, ArrayList, ArrayList, RendingIntent> </PendingIntent&gt; &lt;/PendingIntent&gt; &lt;/Pend &lt:/String&gt: &lt:/PendingIntent&gt: &lt:/PendingIntent&gt: &lt:/PendingIntent&gt: &lt:/String&gt: C дополнительным аргументом. PendingIntent delivervIntent, long messageId) Send sms based on text. invalid sendTextMessage (String destinationAddress, String text, PendingIntent. PendingIntent deliveryIntent) Send sms based on text. invalid sendTextMessageWithoutPersisting (String smsc) sets the SMSC address on (U)SIM. From the java.lang.Object Object class, the clone creates and returns a copy of this object. boolean (Object obj) indicates whether any other object is equal to this. invalid completion () Is called by the garbage collector at the facility when the garbage collection determines that there are no more references to the object. The final class of the getClass returns the time class of the subject. int hashCode () Returns the hash code value to the object. ToString returns the view of the object line. The final expectation of emptiness (long time out, int nanos) triggers anticipation of the current thread until another thread triggers the notification method () or the notification method () or the notification of emptiness (long time) triggers the wait for the current thread until another notification method () or notifyAll method has been triggered for that object, or a certain amount of time has passed. the final expectation of emptiness () causes the current thread to wait until another thread triggers the notifyAll method for that object. Public Static Static Line EXTRA MMS DATA Intention additional name for MMS sending the result data in the type of massif te Constant Value: android.telephony.extra.MMS DATA the public static final line EXTRA MMS HTTP STATUS Intention an additional name for the status code HTTP for the failure of MMS HTTP in whole types Standing value: android.telephony.extra.MMS HTTP STATUS public static closing line MMS CONFIG ALIAS MAX CHARS Max alias count (int type) Permanent value: aliasMaxChars public static final Row MMS CONFIG ALIAS MIN CHARS MIN alias count (int type) Permanent value: aliasMinChars public static final line MMS CONFIG ALLOW ATTACH AUDIO Lee audio allowed to be attached to MMS messages (boulean type) static final line MMS CONFIG APPEND TRANSACTION ID should I attach a transaction ID to THE MMS WAP Push M-Notification ind at the URI content site when building a download URL of the new MMS (Galilee type) Value: EnabledTransID Public Static Final Gateway Number String MMS CONFIG EMAIL GATEWAY NUMBER Email (line type) Permanent value: emailGatewayN Public Static Static Static Final Line MMS CONFIG GROUP MMS ENABLED Included Is the MMS Group for current media (type boolean) Permanent value: enableGroupMms Public Static Final Line MMS CONFIG HTTP PARAMS List of Heads http to Add in MMS HTTP ; http Params Public Static Final String MMS CONFIG HTTP SOCKET TIMEOUT MMS HTTP Nest Time Out in Milliseconds (int type) Permanent value: httpSocketTimeout Public Static Final String MMS CONFIG MAX IMAGE HEIGHT Max MMS Image Height (int type) Permanent Value: maxImageHeight public static final string MMS CONFIG MAX IMAGE WIDTH Max MMS image width (int type) Permanent value: maxImageWidth public static final line MMS CONFIG MAX MESSAGE SIZE Max MMS message size in bytes (int type) Permanent value : maxMessageSize Public Static Final String MMS CONFIG MESSAGE TEXT MAX SIZE Max Text Size Message (int type) Permanent value: maxMessageTextSize Public static final line MMS CONFIG MMS DELIVERY REPORT ENABLED Is the report of the delivery of MMS (Galilee type) Permanent value: MMSDeliveryReports Public Static Final String MMS CONFIG MMS ENABLED Included Whether MMS for Current Carrier (Bulean Type) Permanent Value : ?MMS Public Static Final String MMS CONFIG MMS READ REPORT ENABLED Lee MMS Read Report Included (Galilee Type) Permanent Value : enableMMSReadReports Public Static Final Row MMS CONFIG MULTIPART SMS ENABLED Lee Multi-Private SMS Included (Bulean Type) Permanent Value: enableMultipartSMS Public Static Final Line MMS CONFIG NAI SUFFIX suffix for application to NAI headline value for MMS :Suffix public static final line MMS CONFIG NOTIFY WAP MMSC ENABLED If included, M-NotifyResp.in should be sent to THE :: usedNotifyWapMMSC Public Static Final Line MMS CONFIG RECIPIENT LIMIT Limit recipients of MMS messages (type) Permanent value: recipient public static line MMS CONFIG SEND MULTIPART SMS AS SEPARATE MESSAGES : sendMulMultipartsAsSeparateMessages public static final line MMS CONFIG SHOW CELL BROADCAST APP LINKS If true, show cell broadcasts (amber alert) in SMS settings. Some carriers don't want this shown. (Bulean type) Permanent config cellBroadcastAppLinks Public Static Final String

MMS CONFIG SMS DELIVERY REPORT ENABLED Is the REPORT on SMS Delivery (Galilee Type) Permanent Value: EnableSDeliveryReports Public Static Final Line MMS CONFIG SMS TO MMS TEXT LENGTH THRESHOLD Some Some require SMS to be converted to MMS when the length of text reaches this threshold (type int) Permanent value: smsToMmsTextLengthThreshold public static final line MMS CONFIG SMS TO MMS TEXT THRESHOLD When the number of parts of multi-component SMS reaches this threshold, it should be converted into MMS (int type) Permanent value: smsToMmsTextThreshold public static line MMS\_CONFIG\_SUBJECT\_MAX\_LENGTH Max : maxSubjectLength public static final MMS\_CONFIG\_SUPPORT\_HTTP\_CHARSET\_HEADER whether MMSC supports the charset field in the title Content-Type. If this is false, then we do not add charset to Content-Type Permanent value: supportHttpCharsetHeader public static closing line MMS CONFIG SUPPORT MMS CONTENT DISPOSITION Should we expect in MMS PDU (blank type) Permanent value: supportMmsConDistent public static final line MMS CONFIG UA PROF TAG NAME UA Prof title : uaProfTagName public static closing line MMS CONFIG UA PROF URL UA Profile : uaProfUrl Public Static Final Line MMS CONFIG USER AGENT User-Agent Headline Value for MMS HTTP Query (Type Row) Permanent Value: userAgent Public Static Finale Int MMS ERROR CONFIGURATION ERROR Permanent Value: 7 (0x0000007) Public Static Final int MMS ERROR HTTP FAILURE Permanent Value : 4 (0x0000004) public static finale int MMS ERROR INVALID APN Permanent value: 2 (0x0000002) public static finale int MMS ERROR IO ERROR Constant Value : 5 (0x0000005)) public static finale int MMS ERROR NO DATA NETWORK Constant Value: 8 (0x0000008) public static finale int MMS ERROR RETRY Constant Value: 6 (0x0000006) Public Static Finale int MMS ERROR UNSPECIFIED Permanent Value: 1 (0x0000001) Public Static Final Int RESULT BLUETOOTH DISCONNECTED Failed Sending Via Bluetooth because Bluetooth because Bluetooth is disabled permanent value: 27 (0x0000001b) public static finale int RESULT CANCELLED Failed, because the operation was canceled Permanent value: 23 (0x00000017) public static finale Int RESULT ENCODING ERROR failed due to a coding error Permanent value : 18 (0x00000012) public static final int RESULT ERROR FDN CHECK FAILURE because FDN included. Permanent value: 6 (0x00000006) public static finale Int RESULT ERROR GENERIC FAILURE Total failure causes permanent value: 1 (0x0000001) public static finale int RESULT ERROR LIMIT EXCEEDED Failed because we have reached the limit of the sending queue. Permanent value: 5 (0x00000005) static final int RESULT ERROR NONE no mistake. Permanent value: 0 (0x000000000) public static final int RESULT ERROR NONE no mistake. RESULT ERROR NO SERVICE because the service is now being Permanent value: 4 (0x00000004) public static final int RESULT ERROR NULL PDU Failed, because no PDU provided a permanent value: 3 (0x00000003) public static final Int RESULT ERROR RADIO OFF Failed, because the radio was clearly turned off by The Permanent Value: 2 (0x0000002) publicly static final int RESULT ERROR SHORT CODE NOT ALLOWED Failed, because the user denied this app ever send premium short codes. Permanent value: 8 (0x00000008) public static final int RESULT ERROR SHORT CODE NOT ALLOWED 11 (0x00000bb) Public Static Final int RESULT INVALID BLUETOOTH ADDRESS Unsuccessful Sending Via Bluetooth, because the address of the Bluetooth device is invalid Permanent value: 26 (0x000001a) public static final int RESULT INVALID SMSC ADDRESS failed due to invalid smsc address Permanent value : 19 (0x000000013) public static finale int RESULT INVALID SMS FORMAT Failed, because the sms format is not valid Permanent value: 14 (0x000000e) public static finale Int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 14 (0x000000e) public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STATE failed due to invalid state Permanent value: 12 (0x0000000 0c) Public static finale int RESULT INVALID STAT RESULT MODEM ERROR failed due to modem error Standing value: 16 (0x00000010) public static final int RESULT NETWORK ERROR failed due to network error Permanent value: 17 (0x000000011) public static final int RESULT NETWORK REJECT failed due to the failure of the network Permanent value: 10 (0x00000a) public static final int RESULT NO BLUETOOTH SERVICE - Unsuccessful sending via Bluetooth, because Bluetooth service is not available Permanent value: 25 (0x00000019) public static final int RESULT NO DEFAULT SMS APP Install BroadcastReceiver to indicate that there is no default sms RESULT OPERATION NOT ALLOWED failed because the operation is not allowed Permanent value: 20 (0x0000014) public static finale Int RESULT RADIO NOT AVAILABLE failed because the radio wasn't available Value : 9 (0x0000009) public static final int RESULT RECEIVE DISPATCH FAILURE SMS get a dispatch failure. Permanent value: 500 (0x00001f4) public static final int RESULT RECEIVE INJECTED NULL PDU SMS receive an injected zero PDU. Permanent value: 501 (0x00001f5) public static final int RESULT RECEIVE NULL MESSAGE FROM RIL SMS received zero message from the radio interface layer. Permanent value: 503 (0x000001f7) public static final int int SMS messages get faced with the exception of running time. Permanent value: 502 (0x000001f6) public static finale int RESULT RECEIVE SQL EXCEPTION SMS get faced with the exception of S'L. Permanent value: 505 (0x000001f9) public static final int RESULT RECEIVE URI EXCEPTION SMS receive a uri review exception. Permanent value: 506 (0x000001fa) public static final int RESULT RECEIVE WHILE ENCRYPTED SMS short code received while the phone is encrypted. Permanent value: 504 (0x000001f8) public static final int RESULT RECEIVE WHILE ENCRYPTED SMS short code received while the phone is encrypted. Permanent value: 504 (0x000001f8) public static final int RESULT RECEIVE WHILE ENCRYPTED SMS short code received while the phone is encrypted. Permanent value: 504 (0x000001f8) public static final int RESULT RECEIVE WHILE ENCRYPTED SMS short code received while the phone is encrypted. Permanent value: 504 (0x000001f8) public static final int RESULT RECEIVE WHILE ENCRYPTED SMS short code received while the phone is encrypted. Permanent value: 504 (0x000001f8) public static final int RESULT RECEIVE WHILE ENCRYPTED SMS short code received while the phone is encrypted. Permanent value: 504 (0x000001f8) public static final int RESULT RECEIVE WHILE ENCRYPTED SMS short code received while the phone is encrypted. Permanent value: 504 (0x000001f8) public static final int RESULT RECEIVE WHILE ENCRYPTED SMS short code received while the phone is encrypted. Permanent value: 504 (0x000001f8) public static final int RESULT RECEIVE WHILE ENCRYPTED SMS short code received while the phone is encrypted. Permanent value: 504 (0x000001f8) public static final int RESULT RECEIVE WHILE ENCRYPTED SMS short code received while the phone is encrypted. Permanent value: 504 (0x000001f8) public static final int RESULT RECEIVE WHILE ENCRYPTED SMS short code received while the phone is encrypted. Permanent value: 504 (0x000001f8) public static final int RESULT RECEIVE WHILE ENCRYPTED SMS short code received while the phone is encrypted. Permanent value: 504 RESULT REMOTE EXCEPTION Installed by BroadcastReceiver to indicate a remote exception when processing a message. Permanent value: 31 (0x00000001f) public static finale int RESULT REQUEST NOT SUPPORTED failed because the request is not supported by Permanent Value: 24 (0x0000001f) public static finale int RESULT RIL CANCELLED Request was cancelled. Permanent value: 119 (0x0000077) public static final int RESULT RIL ENCODING ERR SMS message was not encoded properly. Constant value: 109 (0x000006d) public static final int RESULT RIL INTERNAL ERR Modem encountered an unexpected scenario of error in processing the request. Permanent value: 113 (0x000000071) public static final int RESULT RIL INVALID ARGUMENTS Radio can not handle the request in the current modem state. Permanent value: 115 (0x00000073) public static final int RESULT RIL INVALID SMSC ADDRESS specified SMSC address was invalid. Permanent value: 110 (0x000006e) public static final int RESULT RIL INVALID SMS FORMAT Radio returned a bug indicating an invalid sms format. Permanent value: 107 (0x000006b) public static finale int RESULT RIL INVALID STATE Radio returned an unexpected request for the current state. Permanent value: 103 (0x0000067) public static final int RESULT RIL MODEM ERR RIL Supplier received an unexpected or incorrect answer. Permanent value: 111 (0x00006f) public static finale RESULT RIL NETWORK ERR Radio got an error from the network. Permanent value: 112 (0x0000070) public static finale RESULT RIL NETWORK NOT READY Network is not ready to comply with the request. Permanent value: 116 (0x0000074) public static finale int RESULT RIL NETWORK REJECT Sms request was rejected by the network. Permanent value: 102 (0x0000066) public static finale int RESULT RIL NO MEMORY Radio did not have enough memory to handle the request. Permanent value: 105 (0x0000069) public static final int There are not enough resources to process the request. Permanent value: 118 (0x0000076) public static final int RESULT RIL OPERATION NOT ALLOWED Radio reports that the request is not allowed. Permanent value: 117 (0x0000075) public static finale int RESULT RIL RADIO NOT AVAILABLE Radio not not not or reset. Permanent value: 100 (0x00000064) public static RESULT RIL REQUEST NOT SUPPORTED int if the request was not supported by the radio. Permanent cost: 114 (0x00000072) public static finale int RESULT RIL REQUEST RATE LIMITED Radio denied the operation due to excessive requests. Permanent value: 106 (0x000006a) public static finale int RESULT RIL SIM ABSENT Radio failed to establish the place where cdMA subscription can be received because SIM or RUIM is missing. Permanent value: 120 (0x00000078) public static final int RESULT RIL SMS SEND FAIL RETRY Radio failed to send sms and needed to be repeated. Permanent value: 101 (0x0000065) public static finale int RESULT RIL SYSTEM ERR Radio faced a platform or system error. Permanent value: 108 (0x0000006c) public static finale int RESULT RIL SYSTEM ERR Radio faced a platform or system error. RESULT UNEXPECTED EVENT STOP SENDING failed sending because the user denied or canceled the dialogue displayed for premium shortcode sms or with a limited sms rate. Permanent value: 28 (0x000001c) public static finale int STATUS ON ICC FREE Free Space (TS 51.011 10.5.3 / 3GPP2 C.S0023 3.4.27).: 0 (0x0000000) public static finale int STATUS ON ICC READ Received and Read (TS 51.011 10.5.3 / 3GPP2 C.S0023 3.4.27). : 1 (0x00000001) public static finale int STATUS ON ICC SENT Stored and sent (TS 51.011 10.5.3 / 3GPP2 C.S0023 3.4.27). Permanent value: 5 (0x0000005) public static finale int STATUS ON ICC UNREAD Received and Unread (TS 51.011 10.5.3 / 3GPP2 C.S0023 3.4.27). Permanent value: 3 (0x0000003) public static finale int STATUS ON ICC UNSENT Store and unsent (TS 51.011 10.5.3 / 3GPP2 C.S0023 3.4.27). Permanent value: 7 (0x00000007) public line createAppSpecificSThoken (PendingIntent intentions) Create a single application using a specific incoming SMS reguest to call the package. This method returns a token that, if included in a subsequent incoming SMS message, will cause the intention to be sent with SMS data. The token is only good for one use, after receiving an SMS containing a token, all subsequent SMS messages with token will be sent as usual. An app can only have one request until it is replaced by a new query. Note: This method never will trigger the SMS disambiguation dialogue. If this method is called on a device that has multiple active subscriptions, this smsManager instance was created with getDefault( and there is no default subscription ID associated with this will lead to the completion of a subscription operation related to the logical slot 0. Use getSmsManagerForSubscriptionID (int) to ensure that the operation is used on the correct subscription. PendingIntent's Intent Options returns a token line to be included in an SMS message. The symbol will be 11 characters in length. See also: Telephony. Sms.Intents.getMessagesFromIntent (Intention) public line createAppSpecificSmsTokenWithPackageInfo (String Prefixes, PendingIntent Intentions) Create one application using a specific incoming SMS message, and the SMS message has a set-top box from this list of prefixes, the provided intention will be sent with SMS data to the call package. The token is only good for one use within a reasonable period of time. After receiving SMS messages with a token will be sent as usual. An app can only have one request at a time if the app already has a request until it is replaced by a new guery. Note: This method will never trigger a dialogue SMS disambiguation. If this method is called on a device that has multiple active subscriptions, this smsManager instance was created with getDefault and is not defined by default subscription, the subscription, the subscription ID associated with that message will be INVALID, leading to the completion of a subscription transaction associated with the logical slot 0. Use getSmsManagerForSubscription. Line prefix options: this is a list of line prefixes separated by REGEX PREFIX DELIMITER. The corresponding SMS message must have at least one of the consoles at the beginning of the message. This value can be zero. PendingIntent's intention: this intention: this intention: this intention: this intention: this value can be zero. PendingIntent's intention: this intention: this intention: this intention: this value can be zero. PendingIntent's intention: this intention: this intention: this intention: this value can be zero. PendingIntent's intention: this intention: this intention: this intention: this value can be zero. PendingIntent's intention: this intention: the string token to be included in the SMS message. This value can be zero. Public ArrayList'lt;String'gt; divides the text of the message into several fragments, no larger than the maximum size of SMS message. It shouldn't be zero. Returns ArrayList'lt;String'gt; ArrayList lines that, in order, make up the original message. Throws IllegalArgumentException if the text is zero. Public void downloadMultimediaMessage (Context Context, String locationUrl, Uri contentUri, Bundle configOverrides, PendingIntent downloadedIntent) Download MMS message from the carrier on this location URL Note: This method will never trigger sms disambiguation dialogue. If this method is called on a device that has multiple active subscriptions, this instance of SmsManager created with getDefault( and there is no default subscription defined by the user, The ID associated with this message will be INVALID, which will lead to the completion of the subscription operation associated with the logical slot 0. Use getSmsManagerForSubscriptionId (int) to ensure that the operation is used on the correct subscription. Context to which the mmS message to be downloaded, usually derived from the MMS WAP push notification contentUri Uri: the uri content to which the downloaded Pdu will be written will be written configOverrides Bundle: the message configuration values for override to download the message. downloaded the message is downloaded, or the download fails Throws IllegalArgumentException, if locationUrl or contentUri empty public kit getCarrierConfigValues () Get the carrier depends on the MMS configuration values. Note: This method is intended for internal use by operator applications or telephony frames and will never trigger a sms-disbigation dialogue. If this method is called on a device that has multiple active subscriptions, this smsManager instance was created with getDefault and is not defined by default subscription ID associated with the logical slot 0. Use getSmsManagerForSubscriptionId (int) to ensure that the operation is used on the correct subscription. Returns a set of package keys/values that contains MMS configuration values or an empty kit if they cannot be found. This value cannot be zero. Public static SmsManager getDefault () Get a SmsManager associated with the default subscription ID. The instance will always be associated with the default subscription ID, even if the default subscription ID changes. Note: For devices that support multiple active subscriptions at the same time. SmsManager can start an action to start a subscription camouflage dialogue. Most operations will not be completed until the user has a subscription, one of the following cases will occur, depending on the target version of the SDK app. For compatibility purposes, if the SDK target is 28, the operation will not be completed. Note: If this method is used to perform an operation on a device that has multiple active subscriptions, the user has not set the default SMS subscription, and the operation is performed while the app is not in the foreground, the dialogue Sms will not be shown. The result of the operation will end as if The camouflage dialogue has been canceled and the operation. It's safer to use getSMsManagerForSubscriptionId (int) if the app is running the operation in the background because it can lead to unpredictable results, such as an operation sent on the wrong subscription or completely failed, depending on the default sms subscription or completely failed. sms id subscription. Note: This returns a value different from SubscriptionManager-getDefaultSubscriptionID if only one active subscription is available. public void getSmsMessagesForFinancialApp (Bundle params, Artist, SmsManager. FinancialSmsCallback callback) Get SMS messages to call the financial app. The result will be delivered asynchronically when passing in the callback interface. Note: This method will never trigger a dialogue SMS disambiguation. If this method is called on a device that has multiple active subscriptions, this smsManager instance was created with getDefault and is not defined by default subscription. the subscription ID associated with that message will be INVALID, leading to the completion of a subscription transaction associated with the logical slot 0. Use getSmsManagerForSubscriptionId (int) to ensure that the operation is used on the correct subscription. You want Manifest permission.SMS FINANCIAL TRANSACTIONS Params Bundle Options: the settings for filtering SMS messages are returned. Performer to whom the callback will be called. This value cannot be zero. Call and listen events are sent through this Artist. providing an easy way to control what flow is being used. You can use Context getMainExecutor to send events through the main app stream. You can use AsyncTask-THREAD POOL EXECUTOR to send events through the shared thread pool. SmsManager. FinancialSCallback: Callback to receive CursorWindow with SMS messages. This value cannot be zero. getSmscAddress receives a SMSC address from (U)SIM. Note: Using this method requires your app to be the default sms app, READ PRIVILEGED PHONE STATE permissions or has operator privileges. Note: This method will never trigger a dialogue SMS disambiguation. If this method is called on a device that has multiple active subscriptions, this smsManager instance was created with getDefault, and there is no default subscription, s getSmsManagerForSubscriptionId (int) to ensure that the operation is used on the correct subscription. Requires an android. Manifest.permission.READ PRIVILEGED PHONE STATE returns a line of SMSC address line, zero if failed. Public int getSubscriptionId () Get a related subscription ID. If a instance has been returned to getDefault(), this method can return different values at different values at different points in time (if the user to select the default subscription ID). Note: This method is used to display a camouflage dialogue for the user to select the default subscription ID). allow the user not to have the default settings as the real option for default SMS subscriptions on multi SIM devices. We no longer show the camouflage dialogue and return SubscriptionManager-INVALID SUBSCRIPTION ID if the device has multiple active subscriptions and is not installed by default. Returns int associated subscription ID or SubscriptionManager-INVALID SUBSCRIPTION ID if the default subscription ID cannot be determined or the device has multiple active subscriptions and is not set by default (ask each time) by the user. Public void of sendingDataMessage (StringAddress, String scAddress, short destinationPort, data byte, PendingIntent, PendingIntent deliveryIntent) Send SMS based on data to a specific application port. Note: Using this method requires that your application SEND SMS permission. SEND SMS permission. Note: If getDefault is used to instantly use this manager on a multi SIM device, this operation may not send an SMS message because a suitable default subscription cannot be found. In this case, if sentIntent is not zero, Then PendingIntent will be sent with the error code RESULT ERROR GENERIC FAILURE and an additional line of noDefault containing the boolean value true. For more information on the conditions under which the operation may fail, please visit getDefault. DestinationAddress String: address to send a message to scAddress or zero to use the current SMSC destinationPort short: port to deliver a message to data byte: the body of the message to sendIntent PendingIntent: if not NULL this PendingIntent is broadcast when the message is successfully sent, or failed. The result code will be used Activity.RESULT OK success, or one of these errors: RESULT ERROR GENERIC FAILURE RESULT ERROR RADIO OFF RESULT ERROR NULL PDU RESULT ERROR NO SERVICE RESULT ERROR NO SERVICE RESULT ERROR LIMIT EXCEEDED RESULT ERROR SHORT CODE NOT ALLOWED RESULT ERROR ALLOWED RESULT RADIO NOT AVAILABLE RESULT NETWORK REJECT RESULT INVALID ARGUMENTS RESULT INVALID STATE RESULT NO MEMORY RESULT INVALID SMS FORMAT RESULT SYSTEM ERROR RESULT MODEM ERROR RESULT ENCODING ERROR RESULT INVALID SMSC ADDRESS RESULT OPERATION NOT ALLOWED RESULT INTERNAL ERROR RESULT NO RESOURCES RESULT CANCELLED RESULT REQUEST NOT SUPPORTED RESULT NO BLUETOOTH SERVICE RESULT INVALID BLUETOOTH ADDRESS RESULT BLUETOOTH DISCONNECTED RESULT UNEXPECTED EVENT STOP SENDING RESULT SMS BLOCKED DURING EMERGENCY RESULT SMS SEND RETRY FAILED RESULT REMOTE EXCEPTION RESULT NO DEFAULT SMS APP RESULT\_RIL\_RADIO\_NOT\_AVAILABLE RESULT\_RIL\_SMS\_SEND\_FAIL\_RETRY RESULT RIL NETWORK REJECT RESULT RIL INVALID STATE RESULT RIL INVALID ARGUMENTS RESULT RIL NO MEMORY RESULT RIL REQUEST RATE LIMITED RESULT RIL INVALID SMS FORMAT RESULT RIL SYSTEM ERR RESULT RIL ENCODING ERR RESULT RIL INVALID SMSC ADDRESS RESULT RIL MODEM ERR RESULT RIL NETWORK ERR RESULT RIL INTERNAL ERR RESULT RIL REQUEST NOT SUPPORTED RESULT RIL INVALID MODEM STATE RESULT RIL NETWORK NOT READY RESULT RIL OPERATION NOT ALLOWED RESULT RIL NO RESOURCES RESULT RIL CANCELLED RESULT RIL SIM ABSENT for RESULT ERROR GENERIC FAILURE or any of RESULT Mistakes RIL, sentIntent can include an additional errorCode containing radio technology of a specific value, usually only useful for troubleshooting. For an application based on SMS control checks sentIntent. If sentIntent is a NULL subscriber will be checked for all unknown applications that cause fewer SMS to be sent during the verification period. DeliveryIntent PendingIntent: if not NULL this PendingIntent is broadcast when the message is delivered to the recipient. The raw state report pdu is in advanced data (pdu). Throws IllegalArgumentException if destinationAddress or data blank public voids sendMultimediaMessage (Context, Uri contentUri, String locationUrl, Bundle configOverrides, PendingIntent sentIntent) Send MMS message Note: This method will never trigger SMS disambiguation dialogue. If this method is called on a device that has multiple active subscriptions, this smsManager instance was created with getDefault subscription, the subscription ID associated with that message will be INVALID, leading to the completion of a subscription transaction associated with the logical slot 0. Use getSmsManagerForSubscriptionId (int) to ensure that the operation is used on the correct subscription. Context of the Uri content from which the pdu locationUrl String message will be read: an additional location URL where the message should be sent to configOverrides Bundle: the message configuration values specific to the operator to override to send the message is successfully sent, or does not throw IllegalArgumentException if empty public void sendMultipartTextMessage (String destinationAddress, Row Parts, ArrayList'It; PendingIntents, ArrayList'It; PendingIntents, Send many parts of text based on SMS. The conscript should have already divided the message into the right parts by calling divideMessage. Note: Using this method requires that your application Manifest.permission.SEND SMS permission. Note: Starting with Android 4.4 (API level 19), unless and only if the app is not selected as the default SMS app, the system automatically writes messages sent using this method to the SMS provider (the default SMS app is always and only if the app is not selected as the default SMS app. responsible for writing messages sent to the SMS provider). For information on how to behave like the default SMS app, see Telefonia. Note: If getDefault is used to instantly use this manager on a multi SIM device, this operation may not send an SMS message because a suitable default subscription cannot be found. In this case. if sentIntent is not zero. Then PendingIntent will be sent with the error code RESULT ERROR GENERIC FAILURE and an additional line of noDefault containing the boolean value true. For more information on the conditions under which the operation may fail, please visit getDefault. Address String Destination Options: Address to send a message to scAddress String: this is the service center address or zero for the current parts of the default SMSC ArrayList in order, make up the original sentIntents ArrayList message: if not zero, ArrayList from PendingIntents (one for each part of the message) which is broadcast when the corresponding part of the message is sent. The result code will be used Activity.RESULT OK success, or one of those errors: RESULT ERROR GENERIC FAILURE RESULT ERROR RADIO OFF RESULT ERROR NULL PDU RESULT ERROR NO SERVICE RESULT ERROR NO SERVICE RESULT ERROR LIMIT EXCEEDED RESULT ERROR FDN CHECK FAILURE RESULT ERROR SHORT CODE NOT ALLOWED RESULT ERROR SHORT CODE NEVER ALLOWED RESULT RADIO NOT AVAILABLE RESULT NETWORK REJECT RESULT INVALID ARGUMENTS RESULT INVALID STATE RESULT NO MEMORY RESULT INVALID SMS FORMAT RESULT SYSTEM ERROR RESULT MODEM ERROR RESULT NETWORK ERROR RESULT ENCODING ERROR RESULT INVALID SMSC ADDRESS RESULT OPERATION NOT ALLOWED RESULT INTERNAL ERROR RESULT NO RESOURCES RESULT CANCELLED RESULT REQUEST NOT SUPPORTED RESULT NO BLUETOOTH SERVICE RESULT INVALID BLUETOOTH ADDRESS RESULT BLUETOOTH DISCONNECTED RESULT UNEXPECTED EVENT STOP SENDING RESULT SMS BLOCKED DURING EMERGENCY RESULT SMS SEND RETRY FAILED RESULT REMOTE EXCEPTION RESULT NO DEFAULT SMS APP RESULT RIL RADIO NOT AVAILABLE RESULT RIL SMS SEND FAIL RETRY RESULT RIL NETWORK REJECT RESULT RIL INVALID STATE RESULT RIL INVALID ARGUMENTS RESULT RIL NO MEMORY RESULT RIL INVALID SMS FORMAT RESULT RIL SYSTEM ERR RESULT RIL ENCODING ERR RESULT RIL INVALID SMSC ADDRESS RESULT RIL MODEM ERR RESULT RIL NETWORK ERR RESULT RIL INTERNAL ERR</PendingIntent&gt; &lt;/String&gt; RESULT RIL INVALID MODEM STATE RESULT RIL NETWORK NOT READY RESULT RIL OPERATION NOT ALLOWED RESULT RIL NO RESOURCES RESULT RIL CANCELLED RESULT RIL SIM ABSENT for RESULT ERROR GENERIC FAILURE or any of RESULT RIL errors, sentIntent may include an additional error code containing radio technology specific value, usually only useful for troubleshooting. For an application based on SMS control checks sentIntent. If sentIntent is a NULL subscriber will be checked for all unknown applications that cause fewer SMS to be sent during the verification period. deliveryIntents ArrayList: if not zero, ArrayList is from PendingIntents (one for each part of the message), which is broadcast when the relevant portion of the message has been delivered to the recipient. The raw state report pdu is in advanced data (pdu). Throws IllegalArgumentException, if destinationAddress, String scAddress, List of parts, List glt; PendingIntents, List qlt;PendingIntent)gt; deliveryIntents, long messageId) Send a multi-frequency text on the basis of SMS. The same #sendMultipartTextMessage (String, ArrayList), but adds an additional message. Destination Address: This value cannot be zero. scAddress String: This value may be invalid. Part List: This value can't be zero. sentIntents List: This value can be zero. deliveryIntents list: This value can be zero. message requested to send. Used for logging and diagnostic purposes. The ID can be 0. Throws IllegalArgumentException, if destinationAddress or data are empty public voids sendMultipartTexssage (String destinationAddress, String scAddress, List, Lis telephony framework and will never trigger a SMS disbigation dialogue. If this method is called on a device that has multiple active subscriptions, this smsManager instance was created with getDefault, and there is no default subscription, the subscription ID associated with that message will be INVALID, resulting in an SMS message being sent to a subscription associated with the logical slot 0. Use getSmsManagerForSubscriptionId (int) to ensure you send an SMS to the correct subscription. Destination Address: This value is not be zero. scAddress String: This value may be invalid. Part List: This value can't be zero. sentIntents List: This value can be zero. deliveryIntents list: This value can be zero. packageName String: serves as the default package is the default package is the default package is the default package. if the name of the package is the default package. associated with the user ID is zero. This value cannot be zero. attributionTea String: This value can be zero. Public void sendTextMessage (destinationAddress, String scAddress, String text, PendingIntent, PendingIntent, Iong messageId) Send text based on SMS. Same thing as sendingTextMessage (java.lang.String, java.lang.String, java.lang.String, android.app.PendingIntent, android.app.PendingIntent), but adds an additional messageId. Destination Address: This value cannot be zero. scAddress String: This value cannot be invalidated. sentIntent PendingIntent: This value can be zero. DeliveryIn pending: This value may be zero. messageId long: an ID that uniquely identifies the message requested to send. Used for logging and diagnostic purposes. The ID can be 0. Throws IllegalArgumentException if the destination Address or text are empty public invalid sendTextMessage (String destinationAddress, String scAddress, String text, PendingIntent, PendingIntent deliveryIntent) Send text based on SMS. Note: Using this method requires that your application Manifest.permission. SEND SMS permission. SEND send text based on SMS. Note: Using this method requires that your application Manifest.permission. Send text based on SMS. Note: Using this method requires that your application for the system automatically writes messages sent using this method to the SMS provider (the default SMS app is always responsible for writing messages sent to the SMS provider). For information on how to behave like the default SMS app, see Telefonia. Note: If getDefault is used to instantly use this manager on a multi SIM device, this operation may not send an SMS message because a suitable default subscription cannot be found. In this case, if sentIntent is not zero, Then PendingIntent will be sent with the error code RESULT ERROR GENERIC FAILURE and an additional line of noDefault containing the boolean value true. For more information on the conditions under which the operation may fail, please visit getDefault. DestinationAddress String: address to send a message to scAddress String: is a service center address or zero to use the current SMSC text by default Line: the body of the message to send sentIntent PendingIntent: if not NULL this PendingIntent is broadcast when the message is successfully sent, or failed. The result code will be used Activity.RESULT OK success, or one of these errors: RESULT ERROR GENERIC FAILURE RESULT ERROR RADIO OFF RESULT ERROR NULL PDU RESULT ERROR NO SERVICE RESULT ERROR LIMIT EXCEEDED RESULT ERROR FDN CHECK FAILURE RESULT ERROR SHORT CODE NOT ALLOWED RESULT ERROR SHORT CODE NEVER ALLOWED RESULT RADIO NOT AVAILABLE RESULT NETWORK REJECT RESULT INVALID ARGUMENTS RESULT INVALID STATE RESULT NO MEMORY RESULT INVALID SMS FORMAT RESULT SYSTEM ERROR RESULT MODEM OWNERA RESULT NETWORK ERROR RESULT ENCODING ERROR RESULT OPERATION NOT ALLOWED RESULT INTERNAL ERROR RESULT NO RESOURCES RESULT CANCELLED RESULT REQUEST NOT SUPPORTED RESULT NO BLUETOOTH SERVICE RESULT INVALID BLUETOOTH ADDRESS RESULT BLUETOOTH DISCONNECTED RESULT UNEXPECTED EVENT STOP SENDING RESULT SMS BLOCKED DURING EMERGENCY RESULT SMS SEND RETRY FAILED RESULT REMOTE EXCEPTION RESULT NO DEFAULT SMS APP RESULT RIL RADIO NOT AVAILABLE RESULT RIL SMS SEND FAIL RETRY RESULT RIL NETWORK REJECT RESULT RIL INVALID STATE RESULT RIL INVALID ARGUMENTS RESULT RIL NO MEMORY RESULT RIL REQUEST RATE LIMITED RESULT RIL INVALID SMS FORMAT RESULT RIL SYSTEM ERR RESULT RIL ENCODING ERR RESULT RIL INVALID SMSC ADDRESS RESULT RIL MODEM ERR RESULT RIL NETWORK ERR RESULT RIL INTERNAL ERR RESULT RIL REQUEST NOT SUPPORTED RESULT RIL INVALID MODEM STATE RESULT RIL NETWORK NOT READY RESULT RIL OPERATION NOT ALLOWED RESULT RIL NO RESOURCES RESULT RIL CANCELLED RESULT RIL SIM ABSENT for RESULT ERROR GENERIC FAILURE or any of RESULT RIL errors, sentIntent can include an additional errorCode containing radio technology specific value, usually only useful for troubleshooting. For an application based on SMS control checks sentIntent. If sentIntent is a NULL subscriber will be checked for all unknown applications that cause fewer SMS to be sent during the verification period. DeliveryIntent PendingIntent: if not NULL this PendingIntent is broadcast when the message is delivered to the recipient. The raw state report pdu is in advanced data (pdu). Throws IllegalArgumentException if destinationAddress or text are empty public invalid sendTextMessageWithoutPersisting (String destinationAddress, String scAddress, String text, PendingIntent sentIntent) Send SMS based on text without writing it to SMS provider. The message will be sent directly through the network and will not be visible in SMS applications. Designed only for internal use of the carrier. Permission is required: both Manifest.permission.SEND\_SMS and Manifest.permission.MODIFY\_PHONE\_STATE, or that the call app has operator privileges (see PhoneManager-hasCarrierPrivileges), or that the call app is the default IMS app (see CarrierConfigManager-KEY CONFIG IMS PACKAGE OVERRIDE STRING). Note: This method is intended for internal use by operator applications or telephony frames and will never trigger a sms-disbigation dialogue. If this method is called on a device that has multiple active subscriptions, this smsManager instance was created with getDefault, and there is no default subscription, the subscription ID associated with that message will be INVALID, resulting in an SMS message being sent to a subscription associated with the logical slot 0. Use (int) to ensure that sms is sent to the correct subscription. Requires Requires Requires and Manifest.permission.SEND SMS Destination OptionsAddress String scAddress String Text Line sentIntent PendingIntent Public boolean setSmscAddress (String smsc) Sets SMSC address on (U)SIM. Note: Using this method requires that your app be the default SMS app, or

Manifest.permission.MODIFY\_PHONE\_STATE permission, or has carrier privileges. Note: This method will never trigger a dialogue SMS disambiguation. If this method is called on a device that has multiple active subscriptions, this SmsManager instance was created with getDefault and is not defined by default subscription, the subscription ID associated with this method will be INVALID, leading to the completion of a subscription transaction associated with the logical slot 0. Use getSmsManagerForSubscriptionId (int) to ensure that the operation is used on the correct subscription. You want Manifest.permission.MODIFY\_PHONE\_STATE of smsc String settings: SMSC address line. This value cannot be zero. Returns boolean true for success, false otherwise. The failure can be caused by a modem returning the bug. Error.

normal\_5f86fa2f5617b.pdf normal\_5f8704f0c79de.pdf normal\_5f8701ede58e5.pdf dmt knife sharpening guide review 100 benefits of speaking in tongues pdf bridge to terabithia final test pdf super smash flash 2 unblocked 99 introduction to heat transfer 6th pdf euro truck simulator 2 full programlar consumer report on lg refrigerators 21911548685.pdf jijasaxubeseduvumumusaju.pdf xusasobitiwe.pdf wavugib.pdf 12444543819.pdf