android.hardware.Camera.Parameters---startFaceDetection:   
Starts the face detection. This should be called after preview is started. The camera will notify Camera.FaceDetectionListener of the detected faces in the preview frame. The detected faces may be the same as the previous ones. Applications should call stopFaceDetection() to stop the face detection. This method is supported if getMaxNumDetectedFaces() returns a number larger than 0. If the face detection has started, apps should not call this again. When the face detection is running, setWhiteBalance(String), setFocusAreas(List), and setMeteringAreas(List) have no effect. The camera uses the detected faces to do auto-white balance, auto exposure, and autofocus. If the apps call autoFocus(AutoFocusCallback), the camera will stop sending face callbacks. The last face callback indicates the areas used to do autofocus. After focus completes, face detection will resume sending face callbacks. If the apps call cancelAutoFocus(), the face callbacks will also resume. After calling takePicture(Camera.ShutterCallback, Camera.PictureCallback, Camera.PictureCallback) or stopPreview(), and then resuming preview with startPreview(), the apps should call this method again to resume face detection. See also: Camera.FaceDetectionListenerstopFaceDetection()getMaxNumDetectedFaces()  
  
android.hardware.camera2.CaptureResult---wasImageCaptured:   
Determine if the image was captured from the camera. If the image was not captured, no image buffers will be available. If the image was captured, then image buffers may be available.  
  
android.hardware.Camera.Parameters---setPreviewCallback:   
Installs a callback to be invoked for every preview frame in addition to displaying them on the screen. The callback will be repeatedly called for as long as preview is active. This method can be called at any time, even while preview is live. Any other preview callbacks are overridden. If you are using the preview data to create video or still images, strongly consider using MediaActionSound to properly indicate image capture or recording start/stop to the user. See also: MediaActionSound  
  
android.media.audiofx.EnvironmentalReverb.Settings---AudioEffect.Descriptor:   
  
  
android.media.MediaDrm.HdcpLevel---getAltitude:   
Get the altitude if available, in meters above the WGS 84 reference ellipsoid. If this location does not have an altitude then 0.0 is returned.  
  
android.widget.RadioButton---update:   
Updates the position and the dimension of the popup window. Width and height can be set to -1 to update location only. Calling this function also updates the window with the current popup state as described for update().  
  
android.renderscript.ScriptIntrinsicConvolve3x3---setLUT:   
Sets the Allocation to be used as the lookup table. The lookup table must use the same Element as the intrinsic.  
  
android.service.carrier.MessagePdu---onFilterSms:   
This method was deprecated in API level 24. Use onReceiveTextSms(MessagePdu, String, int, int, CarrierMessagingService.ResultCallback) instead. Override this method to filter inbound SMS messages.  
  
android.location.GnssMeasurement---setBearingAccuracy:   
Indicates the desired bearing accuracy. Accuracy may be ACCURACY\_LOW, ACCURACY\_HIGH, or NO\_REQUIREMENT. More accurate location may consume more power and may take longer.  
  
android.bluetooth.BluetoothGattServer---onDescriptorRead:   
Callback reporting the result of a descriptor read operation.  
  
android.webkit.WebMessage---getCurrentIndex:   
Get the index of the current history item. This index can be used to directly index into the array list.  
  
android.test.ComparisonFailure---drag:   
This method was deprecated in API level 3. ActivityInstrumentationTestCase is deprecated in favor of ActivityInstrumentationTestCase2, which provides more options for configuring the Activity under test Simulate touching a specific location and dragging to a new location.  
  
android.util.SparseLongArray---SparseArray:   
Creates a new SparseArray containing no mappings that will not require any additional memory allocation to store the specified number of mappings. If you supply an initial capacity of 0, the sparse array will be initialized with a light-weight representation not requiring any additional array allocations.  
  
android.util.StringBuilderPrinter---SparseLongArray:   
Creates a new SparseLongArray containing no mappings that will not require any additional memory allocation to store the specified number of mappings. If you supply an initial capacity of 0, the sparse array will be initialized with a light-weight representation not requiring any additional array allocations.  
  
android.net.wifi.p2p.nsd.WifiP2pServiceInfo---setDnsSdResponseListeners:   
Register a callback to be invoked on receiving Bonjour service discovery response. see discoverServices(WifiP2pManager.Channel, WifiP2pManager.ActionListener) for the detail.  
  
android.telecom.TelecomManager---requestCameraCapabilities:   
Issues a request to retrieve the capabilities of the current camera for the RemoteConnection.VideoProvider. See also: onRequestCameraCapabilities()  
  
android.bluetooth.BluetoothClass.Device.Major---closeProfileProxy:   
Close the connection of the profile proxy to the Service. Clients should call this when they are no longer using the proxy obtained from getProfileProxy(Context, BluetoothProfile.ServiceListener, int). Profile can be one of HEALTH, HEADSET or A2DP  
  
android.os.strictmode.CustomViolation---getAllocatableBytes:   
Return the maximum number of new bytes that your app can allocate for itself on the given storage volume. This value is typically larger than getUsableSpace(), since the system may be willing to delete cached files to satisfy an allocation request. You can then allocate space for yourself using allocateBytes(UUID, long) or allocateBytes(FileDescriptor, long). This method is best used as a pre-flight check, such as deciding if there is enough space to store an entire music album before you allocate space for each audio file in the album. Attempts to allocate disk space beyond the returned value will fail. If the returned value is not large enough for the data you'd like to persist, you can launch ACTION\_MANAGE\_STORAGE with the EXTRA\_UUID and EXTRA\_REQUESTED\_BYTES options to help involve the user in freeing up disk space. If you're progressively allocating an unbounded amount of storage space (such as when recording a video) you should avoid calling this method more than once every 30 seconds. Note: if your app uses the android:sharedUserId manifest feature, then allocatable space for all packages in your shared UID is tracked together as a single unit. This method may take several seconds to complete, so it should only be called from a worker thread. Value is a non-negative number of bytes.  
  
android.preference.TwoStatePreference---isStorageDeviceProtected:   
Indicates if the storage location used internally by this class is backed by device-protected storage. See also: setStorageDefault()setStorageDeviceProtected()  
  
android.test.ComparisonFailure---dragViewToY:   
Simulate touching a view and dragging it to a specified location. Only moves vertically.  
  
android.bluetooth.BluetoothHidDeviceAppQosSettings---connectChannelToSource:   
Connect to a health device which has the SOURCE\_ROLE. This is an asynchronous call. If this function returns true, the callback associated with the application configuration will be called. Requires BLUETOOTH permission.  
  
android.app.backup.BackupHelper---setWebUri:   
Set a web URI associated with the current data being shown to the user. This URI could be opened in a web browser, or in the app as an ACTION\_VIEW Intent, to show the same data that is currently being displayed by it. The URI here should be something that is transportable off the device into other environments to acesss the same data as is currently being shown in the app; if the app does not have such a representation, it should leave the null and only report the local intent and clip data.  
  
android.security.keystore.KeyExpiredException---isInvalidatedByBiometricEnrollment:   
Returns true if the key is irreversibly invalidated when a new fingerprint is enrolled or all enrolled fingerprints are removed. This has effect only for keys that require fingerprint user authentication for every use. See also: isUserAuthenticationRequired()getUserAuthenticationValidityDurationSeconds()setInvalidatedByBiometricEnrollment(boolean)  
  
android.media.effect.EffectFactory---onChildrenLoaded:   
Called when the list of children is loaded or updated.  
  
java.util.zip.GZIPOutputStream---setDictionary:   
Sets preset dictionary for compression. A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with Inflater.inflate(), Inflater.getAdler() can be called in order to get the Adler-32 value of the dictionary required for decompression. See also: inflate(byte[])getAdler()  
  
android.util.StatsLog---SparseIntArray:   
Creates a new SparseIntArray containing no mappings that will not require any additional memory allocation to store the specified number of mappings. If you supply an initial capacity of 0, the sparse array will be initialized with a light-weight representation not requiring any additional array allocations.  
  
android.nfc.NfcManager---NdefRecord:   
Construct an NDEF Record from its component fields. Recommend to use helpers such as {#createUri} or {createExternal(String, String, byte[]) where possible, since they perform stricter validation that the record is correctly formatted as per NDEF specifications. However if you know what you are doing then this constructor offers the most flexibility. An NdefRecord represents a logical (complete) record, and cannot represent NDEF Record chunks. Basic validation of the tnf, type, id and payload is performed as per the following rules: The tnf paramter must be a 3-bit value. Records with a tnf of TNF\_EMPTY cannot have a type, id or payload. Records with a tnf of TNF\_UNKNOWN or 0x07 cannot have a type. Records with a tnf of TNF\_UNCHANGED are not allowed since this class only represents complete (unchunked) records. This minimal validation is specified by NFCForum-TS-NDEF\_1.0 section 3.2.6 (Type Name Format). If any of the above validation steps fail then IllegalArgumentException is thrown. Deep inspection of the type, id and payload fields is not performed, so it is possible to create NDEF Records that conform to section 3.2.6 but fail other more strict NDEF specification requirements. For example, the payload may be invalid given the tnf and type. To omit a type, id or payload field, set the parameter to an empty byte array or null.  
  
android.telephony.SubscriptionManager.OnSubscriptionsChangedListener---getTPLayerLengthForPDU:   
Get the TP-Layer-Length for the given SMS-SUBMIT PDU Basically, the length in bytes (not hex chars) less the SMSC header *FIXME: This method is only used by a CTS test case that isn't run on CDMA devices. We should probably deprecate it and remove the obsolete test case.*android.net.wifi.p2p.WifiP2pDeviceList---onServiceAvailable:   
The requested service response is available.  
  
android.telephony.SignalStrength---formatNumber:   
This method was deprecated in API level 21. Use link #formatNumber(String phoneNumber, String defaultCountryIso) instead Breaks the given number down and formats it according to the rules for the country the number is from.  
  
android.hardware.GeomagneticField---getFocusDistances:   
Gets the distances from the camera to where an object appears to be in focus. The object is sharpest at the optimal focus distance. The depth of field is the far focus distance minus near focus distance. Focus distances may change after calling autoFocus(AutoFocusCallback), cancelAutoFocus(), or startPreview(). Applications can call getParameters() and this method anytime to get the latest focus distances. If the focus mode is FOCUS\_MODE\_CONTINUOUS\_VIDEO, focus distances may change from time to time. This method is intended to estimate the distance between the camera and the subject. After autofocus, the subject distance may be within near and far focus distance. However, the precision depends on the camera hardware, autofocus algorithm, the focus area, and the scene. The error can be large and it should be only used as a reference. Far focus distance >= optimal focus distance >= near focus distance. If the focus distance is infinity, the value will be Float.POSITIVE\_INFINITY. See also: FOCUS\_DISTANCE\_NEAR\_INDEXFOCUS\_DISTANCE\_OPTIMAL\_INDEXFOCUS\_DISTANCE\_FAR\_INDEX  
  
android.renderscript.ScriptIntrinsicConvolve5x5---forEachClear:   
Sets dst = {0, 0, 0, 0}  
  
android.opengl.GLES32.DebugProc---setTimeout:   
Set the transceive(byte[]) timeout in milliseconds. The timeout only applies to transceive(byte[]) on this object, and is reset to a default value when close() is called. Setting a longer timeout may be useful when performing transactions that require a long processing time on the tag such as key generation. Requires the NFC permission.  
  
android.net.UrlQuerySanitizer.ValueSanitizer---getSequenceNumberLong:   
Returns the sequence number for the MTP object This field is typically not used for MTP devices, but is sometimes used to define a sequence of photos on PTP cameras.  
  
android.media.MediaDrm.SecurityLevel---unregisterGnssMeasurementsCallback:   
Unregisters a GPS Measurement callback.  
  
android.net.wifi.p2p.nsd.WifiP2pServiceInfo---connect:   
Start a p2p connection to a device with the specified configuration. The function call immediately returns after sending a connection request to the framework. The application is notified of a success or failure to initiate connect through listener callbacks onSuccess() or onFailure(int). Register for WIFI\_P2P\_CONNECTION\_CHANGED\_ACTION intent to determine when the framework notifies of a change in connectivity. If the current device is not part of a p2p group, a connect request initiates a group negotiation with the peer. If the current device is part of an existing p2p group or has created a p2p group with createGroup(WifiP2pManager.Channel, WifiP2pManager.ActionListener), an invitation to join the group is sent to the peer device.  
  
android.hardware.GeomagneticField---setGpsProcessingMethod:   
Sets GPS processing method. The method will be stored in a UTF-8 string up to 31 bytes long, in the JPEG EXIF header.  
  
android.media.MediaDrm.HdcpLevel---reset:   
Clears the contents of the location.  
  
android.net.sip.SipSession.Listener---isSipWifiOnly:   
Returns true if SIP is only available on WIFI.  
  
android.hardware.camera2.CameraDevice---setRepeatingRequest:   
Request endlessly repeating capture of images by this capture session. With this method, the camera device will continually capture images using the settings in the provided CaptureRequest, at the maximum rate possible. Repeating requests are a simple way for an application to maintain a preview or other continuous stream of frames, without having to continually submit identical requests through capture(CaptureRequest, CameraCaptureSession.CaptureCallback, Handler). Repeat requests have lower priority than those submitted through capture(CaptureRequest, CameraCaptureSession.CaptureCallback, Handler) or captureBurst(List, CameraCaptureSession.CaptureCallback, Handler), so if capture(CaptureRequest, CameraCaptureSession.CaptureCallback, Handler) is called when a repeating request is active, the capture request will be processed before any further repeating requests are processed. To stop the repeating capture, call stopRepeating(). Calling abortCaptures() will also clear the request. Calling this method will replace any earlier repeating request or burst set up by this method or setRepeatingBurst(List, CameraCaptureSession.CaptureCallback, Handler), although any in-progress burst will be completed before the new repeat request will be used. This method does not support reprocess capture requests because each reprocess CaptureRequest must be created from the TotalCaptureResult that matches the input image to be reprocessed. This is either the TotalCaptureResult of capture that is sent for reprocessing, or one of the TotalCaptureResults of a set of captures, when data from the whole set is combined by the application into a single reprocess input image. The request must be capturing images from the camera. If a reprocess capture request is submitted, this method will throw IllegalArgumentException. See also: capture(CaptureRequest, CameraCaptureSession.CaptureCallback, Handler)captureBurst(List, CameraCaptureSession.CaptureCallback, Handler)setRepeatingBurst(List, CameraCaptureSession.CaptureCallback, Handler)stopRepeating()abortCaptures()  
  
android.hardware.GeomagneticField---setPreviewFormat:   
Sets the image format for preview pictures. If this is never called, the default format will be NV21, which uses the NV21 encoding format. Use getSupportedPreviewFormats() to get a list of the available preview formats. It is strongly recommended that either NV21 or YV12 is used, since they are supported by all camera devices. For YV12, the image buffer that is received is not necessarily tightly packed, as there may be padding at the end of each row of pixel data, as described in YV12. For camera callback data, it can be assumed that the stride of the Y and UV data is the smallest possible that meets the alignment requirements. That is, if the preview size is width x height, then the following equations describe the buffer index for the beginning of row y for the Y plane and row c for the U and V planes: yStride = (int) ceil(width / 16.0) \* 16; uvStride = (int) ceil( (yStride / 2) / 16.0) \* 16; ySize = yStride \* height; uvSize = uvStride \* height / 2; yRowIndex = yStride \* y; uRowIndex = ySize + uvSize + uvStride \* c; vRowIndex = ySize + uvStride \* c; size = ySize + uvSize \* 2; See also: ImageFormatgetSupportedPreviewFormats()  
  
android.nfc.tech.NfcB---writePage:   
Write 1 page (4 bytes). The MIFARE Ultranlight protocol always writes 1 page at a time, to minimize EEPROM write cycles. This is an I/O operation and will block until complete. It must not be called from the main application thread. A blocked call will be canceled with IOException if close() is called from aother thread. Requires the NFC permission.  
  
android.telecom.PhoneAccount.Builder---requestBluetoothAudio:   
Request audio routing to a specific bluetooth device. Calling this method may result in the device routing audio to a different bluetooth device than the one specified if the bluetooth stack is unable to route audio to the requested device. A list of available devices can be obtained via getSupportedBluetoothDevices()  
  
android.renderscript.ScriptIntrinsicConvolve5x5---forEachSrcOver:   
Sets dst = src + dst \* (1.0 - src.a)  
  
android.hardware.SensorListener---onPictureTaken:   
Called when image data is available after a picture is taken. The format of the data depends on the context of the callback and Camera.Parameters settings.  
  
android.location.Geocoder---onProviderEnabled:   
Called when the provider is enabled by the user.  
  
android.test.mock.MockAccountManager---assertBottomAligned:   
Assert that two views are bottom aligned, that is that their bottom edges are on the same y location, with respect to the specified margin.  
  
android.accounts.AccountManagerCallback---unregisterFingerprintGestureCallback:   
Unregister a listener added with registerFingerprintGestureCallback(FingerprintGestureController.FingerprintGestureCallback, Handler).  
  
android.net.wifi.p2p.nsd.WifiP2pServiceInfo---cancelConnect:   
Cancel any ongoing p2p group negotiation The function call immediately returns after sending a connection cancellation request to the framework. The application is notified of a success or failure to initiate cancellation through listener callbacks onSuccess() or onFailure(int).  
  
android.media.session.MediaSessionManager---onPrepareFromMediaId:   
Override to handle requests to prepare for playing a specific mediaId that was provided by your app's MediaBrowserService. During the preparation, a session should not hold audio focus in order to allow other sessions play seamlessly. The state of playback should be updated to STATE\_PAUSED after the preparation is done. The playback of the prepared content should start in the implementation of onPlay(). Override onPlayFromMediaId(String, Bundle) to handle requests for starting playback without preparation.  
  
android.bluetooth.le.PeriodicAdvertisingParameters.Builder---startAdvertising:   
Start Bluetooth LE Advertising. On success, the advertiseData will be broadcasted. Returns immediately, the operation status is delivered through callback. Requires BLUETOOTH\_ADMIN permission.  
  
android.media.session.MediaSessionManager---onPlayFromMediaId:   
Override to handle requests to play a specific mediaId that was provided by your app's MediaBrowserService.  
  
android.renderscript.ScriptIntrinsicConvolve5x5---forEachDstIn:   
Sets dst = dst \* src.a  
  
java.time.Month---isSupported:   
Checks if the specified unit is supported. This checks if the specified unit can be added to, or subtracted from, this date-time. If false, then calling the plus(long, TemporalUnit) and minus methods will throw an exception. If the unit is a ChronoUnit then the query is implemented here. The supported units are: NANOS MICROS MILLIS SECONDS MINUTES HOURS HALF\_DAYS DAYS WEEKS MONTHS YEARS DECADES CENTURIES MILLENNIA ERAS All other ChronoUnit instances will return false. If the unit is not a ChronoUnit, then the result of this method is obtained by invoking TemporalUnit.isSupportedBy(Temporal) passing this as the argument. Whether the unit is supported is determined by the unit.  
  
android.bluetooth.BluetoothClass.Device.Major---getScanMode:   
Get the current Bluetooth scan mode of the local Bluetooth adapter. The Bluetooth scan mode determines if the local adapter is connectable and/or discoverable from remote Bluetooth devices. Possible values are: SCAN\_MODE\_NONE, SCAN\_MODE\_CONNECTABLE, SCAN\_MODE\_CONNECTABLE\_DISCOVERABLE. If Bluetooth state is not STATE\_ON, this API will return SCAN\_MODE\_NONE. After turning on Bluetooth, wait for ACTION\_STATE\_CHANGED with STATE\_ON to get the updated value. Requires the BLUETOOTH permission.  
  
android.telephony.AccessNetworkConstants.GeranBand---VideoProfile.CameraCapabilities:   
Create a call camera capabilities instance.  
  
android.graphics.drawable.GradientDrawable---setHotspot:   
Specifies the hotspot's location within the drawable.  
  
android.webkit.PermissionRequest---getOrigins:   
Gets the set of origins for which Geolocation permissions are stored.  
  
android.graphics.ColorFilter---getLocationZ:   
Gets the z location of the camera. See also: setLocation(float, float, float)  
  
android.telephony.mbms.DownloadStateCallback---calculateLength:   
This method was deprecated in API level 4. Use android.telephony.SmsMessage. Calculates the number of SMS's required to encode the message body and the number of characters remaining until the next message, given the current encoding.  
  
android.media.MediaDrm.HdcpLevel---removeBearing:   
This method was deprecated in API level 26. use a new Location object for location updates. Remove the bearing from this location. Following this call hasBearing() will return false, and getBearing() will return 0.0.  
  
android.appwidget.AppWidgetManager---getAppStandbyBucket:   
Returns the current standby bucket of the calling app. The system determines the standby state of the app based on app usage patterns. Standby buckets determine how much an app will be restricted from running background tasks such as jobs and alarms. Restrictions increase progressively from STANDBY\_BUCKET\_ACTIVE to STANDBY\_BUCKET\_RARE, with STANDBY\_BUCKET\_ACTIVE being the least restrictive. The battery level of the device might also affect the restrictions. Apps in buckets �� STANDBY\_BUCKET\_ACTIVE have no standby restrictions imposed. Apps in buckets > STANDBY\_BUCKET\_FREQUENT may have network access restricted when running in the background. The standby state of an app can change at any time either due to a user interaction or a system interaction or some algorithm determining that the app can be restricted for a period of time before the user has a need for it. You can also query the recent history of standby bucket changes by calling queryEventsForSelf(long, long) and searching for STANDBY\_BUCKET\_CHANGED.  
  
android.test.mock.MockAccountManager---assertLeftAligned:   
Assert that two views are left aligned, that is that their left edges are on the same x location, with respect to the specified margin.  
  
android.nfc.tech.NdefFormatable---transceive:   
Send raw ISO-DEP data to the tag and receive the response. Applications must only send the INF payload, and not the start of frame and end of frame indicators. Applications do not need to fragment the payload, it will be automatically fragmented and defragmented by transceive(byte[]) if it exceeds FSD/FSC limits. Use getMaxTransceiveLength() to retrieve the maximum number of bytes that can be sent with transceive(byte[]). This is an I/O operation and will block until complete. It must not be called from the main application thread. A blocked call will be canceled with IOException if close() is called from another thread. Requires the NFC permission.  
  
android.graphics.ColorFilter---getLocationY:   
Gets the y location of the camera. See also: setLocation(float, float, float)  
  
android.hardware.camera2.CameraDevice.StateCallback---onCaptureSequenceCompleted:   
This method is called independently of the others in CaptureCallback, when a capture sequence finishes and all CaptureResult or CaptureFailure for it have been returned via this listener. In total, there will be at least one result/failure returned by this listener before this callback is invoked. If the capture sequence is aborted before any requests have been processed, onCaptureSequenceAborted(CameraCaptureSession, int) is invoked instead. The default implementation does nothing. See also: getFrameNumber()getFrameNumber()getSequenceId()getSequenceId()onCaptureSequenceAborted(CameraCaptureSession, int)  
  
android.telephony.CellSignalStrengthWcdma---requestLocationUpdate:   
Request an update of the current location. If the location has changed, a broadcast will be sent to everyone registered with LISTEN\_CELL\_LOCATION.  
  
android.bluetooth.le.ScanResult---onScanResult:   
Callback when a BLE advertisement has been found.  
  
android.telecom.InCallService---ConnectionRequest:   
  
  
android.media.MediaDrm.SecurityLevel---addGpsStatusListener:   
This method was deprecated in API level 24. use registerGnssStatusCallback(GnssStatus.Callback) instead. Adds a GPS status listener. Requires the ACCESS\_FINE\_LOCATION permission.  
  
android.os.strictmode.CustomViolation---allocateBytes:   
Allocate the requested number of bytes for your application to use in the given open file. This will cause the system to delete any cached files necessary to satisfy your request. Attempts to allocate disk space beyond the value returned by getAllocatableBytes(UUID) will fail. This method guarantees that bytes have been allocated to the opened file, otherwise it will throw if fast allocation is not possible. Fast allocation is typically only supported in private app data directories, and on shared/external storage devices which are emulated. If you're progressively allocating an unbounded amount of storage space (such as when recording a video) you should avoid calling this method more than once every 60 seconds. This method may take several seconds to complete, so it should only be called from a worker thread. See also: ERROR(/#getAllocatableBytes(UUID, int))isAllocationSupported(FileDescriptor)isExternalStorageEmulated(File)  
  
android.media.MediaDrm.HdcpLevel---removeAltitude:   
This method was deprecated in API level 26. use a new Location object for location updates. Remove the altitude from this location. Following this call hasAltitude() will return false, and getAltitude() will return 0.0.  
  
android.test.UiThreadTest---getLac:   
  
  
android.preference.TwoStatePreference---isStorageDefault:   
Indicates if the storage location used internally by this class is the default provided by the hosting Context. See also: setStorageDefault()setStorageDeviceProtected()  
  
android.widget.RadioButton---showAsDropDown:   
Displays the content view in a popup window anchored to the corner of another view. The window is positioned according to the specified gravity and offset by the specified x and y coordinates. If there is not enough room on screen to show the popup in its entirety, this method tries to find a parent scroll view to scroll. If no parent view can be scrolled, the specified vertical gravity will be ignored and the popup will anchor itself such that it is visible. If the view later scrolls to move anchor to a different location, the popup will be moved correspondingly. See also: dismiss()  
  
android.renderscript.ScriptIntrinsicConvolve5x5---forEachSrcOver:   
Sets dst = src + dst \* (1.0 - src.a)  
  
android.hardware.GeomagneticField---setGpsLatitude:   
Sets GPS latitude coordinate. This will be stored in JPEG EXIF header.  
  
android.hardware.camera2.CameraManager.TorchCallback---onError:   
The method called when a camera device has encountered a serious error. This callback may be called instead of onOpened(CameraDevice) if opening the camera fails. This indicates a failure of the camera device or camera service in some way. Any attempt to call methods on this CameraDevice in the future will throw a CameraAccessException with the CAMERA\_ERROR reason. There may still be capture completion or camera stream callbacks that will be called after this error is received. You should clean up the camera with close() after this happens. Further attempts at recovery are error-code specific. See also: ERROR\_CAMERA\_IN\_USEERROR\_MAX\_CAMERAS\_IN\_USEERROR\_CAMERA\_DISABLEDERROR\_CAMERA\_DEVICEERROR\_CAMERA\_SERVICE  
  
android.widget.RadioGroup---assignContactFromPhone:   
Assign a contact based on a phone number. This should only be used when the contact's URI is not available, as an extra query will have to be performed to lookup the URI based on the phone number.  
  
android.nfc.tech.NfcA---getTimeout:   
Get the current transceive(byte[]) timeout in milliseconds. Requires the NFC permission.  
  
android.net.wifi.p2p.WifiP2pManager.ServiceResponseListener---onConnectionInfoAvailable:   
The requested connection info is available  
  
android.net.Uri.Builder---SSLSessionCache:   
Create a session cache at the default location for this app. Multiple instances share data internally.  
  
android.location.GpsSatellite---getMessageId:   
Gets the Message identifier. This provides an index to help with complete Navigation Message assembly. Similar identifiers within the data bits themselves often supplement this information, in ways even more specific to each message type; see the relevant satellite constellation ICDs for details. For GPS L1 C/A subframe 4 and 5, this value corresponds to the 'frame id' of the navigation message, in the range of 1-25 (Subframe 1, 2, 3 does not contain a 'frame id' and this value can be set to -1.) For Glonass L1 C/A, this refers to the frame ID, in the range of 1-5. For BeiDou D1, this refers to the frame number in the range of 1-24 For Beidou D2, this refers to the frame number, in the range of 1-120 For Galileo F/NAV nominal frame structure, this refers to the subframe number, in the range of 1-12 For Galileo I/NAV nominal frame structure, this refers to the subframe number in the range of 1-24  
  
android.telecom.RemoteConnection---setCallAudioState:   
Request to change the conference's audio routing to the specified state. The specified state can include audio routing (Bluetooth, Speaker, etc) and muting state.  
  
android.nfc.tech.NfcF---format:   
Format a tag as NDEF, and write a NdefMessage. This is a multi-step process, an IOException is thrown if any one step fails. The card is left in a read-write state after this operation. This is an I/O operation and will block until complete. It must not be called from the main application thread. A blocked call will be canceled with IOException if close() is called from another thread. Requires the NFC permission.  
  
android.hardware.Camera.Parameters---stopSmoothZoom:   
Stops the smooth zoom. Applications should wait for the Camera.OnZoomChangeListener to know when the zoom is actually stopped. This method is supported if isSmoothZoomSupported() is true.  
  
android.nfc.tech.TagTechnology---processCommandApdu:   
This method will be called when a command APDU has been received from a remote device. A response APDU can be provided directly by returning a byte-array in this method. Note that in general response APDUs must be sent as quickly as possible, given the fact that the user is likely holding his device over an NFC reader when this method is called. If there are multiple services that have registered for the same AIDs in their meta-data entry, you will only get called if the user has explicitly selected your service, either as a default or just for the next tap. This method is running on the main thread of your application. If you cannot return a response APDU immediately, return null and use the sendResponseApdu(byte[]) method later.  
  
android.nfc.cardemulation.OffHostApduService---isDefaultServiceForAid:   
Allows an application to query whether a service is currently the default handler for a specified ISO7816-4 Application ID.  
  
android.nfc.Tag---disableForegroundNdefPush:   
This method was deprecated in API level 14. use setNdefPushMessage(NdefMessage, Activity, Activity...) instead Disable NDEF message push over P2P. After calling enableForegroundNdefPush(Activity, NdefMessage), an activity must call this method before its onPause() callback completes. Strongly recommend to use the new setNdefPushMessage(NdefMessage, Activity, Activity...) instead: it automatically hooks into your activity life-cycle, so you do not need to call enable/disable in your onResume/onPause. This method must be called from the main thread. Requires the NFC permission.  
  
android.provider.ContactsContract.RawContacts.DisplayPhoto---showQuickContact:   
Trigger a dialog that lists the various methods of interacting with the requested Contacts entry. This may be based on available ContactsContract.Data rows under that contact, and may also include social status and presence details.  
  
android.test.ComparisonFailure---dragViewTo:   
Simulate touching a view and dragging it to a specified location.  
  
android.bluetooth.BluetoothSocket---BluetoothHidDeviceAppSdpSettings:   
Create a BluetoothHidDeviceAppSdpSettings object for the Bluetooth SDP record.  
  
android.service.notification.Condition---notifyChildrenChanged:   
Notifies all connected media browsers that the children of the specified parent id have changed in some way. This will cause browsers to fetch subscribed content again.  
  
android.test.UiThreadTest---setStateInvalid:   
Invalidate this object. The location area code and the cell id are set to -1.  
  
android.bluetooth.BluetoothGattServer---onConnectionStateChange:   
Callback indicating when GATT client has connected/disconnected to/from a remote GATT server.  
  
android.net.wifi.WpsInfo---release:   
Unlocks the Wi-Fi radio, allowing it to turn off when the device is idle. If this WifiLock is reference-counted, each call to release will decrement the reference count, and the radio will be unlocked only when the reference count reaches zero. If the reference count goes below zero (that is, if release is called a greater number of times than acquire()), an exception is thrown. If this WifiLock is not reference-counted, the first call to release (after the radio was locked using acquire()) will unlock the radio, and subsequent calls will be ignored.  
  
android.media.MediaDrm.HdcpLevel---hasSpeed:   
True if this location has a speed.  
  
android.bluetooth.BluetoothGattServer---onCharacteristicWrite:   
Callback indicating the result of a characteristic write operation. If this callback is invoked while a reliable write transaction is in progress, the value of the characteristic represents the value reported by the remote device. An application should compare this value to the desired value to be written. If the values don't match, the application must abort the reliable write transaction.  
  
android.hardware.camera2.CameraDevice---captureBurstRequests:   
Submit a list of requests to be captured in sequence as a burst. The burst will be captured in the minimum amount of time possible, and will not be interleaved with requests submitted by other capture or repeat calls. The behavior of this method matches that of captureBurst(List, CaptureCallback, Handler), except that it uses Executor as an argument instead of Handler. See also: capture(CaptureRequest, CameraCaptureSession.CaptureCallback, Handler)setRepeatingRequest(CaptureRequest, CameraCaptureSession.CaptureCallback, Handler)setRepeatingBurst(List, CameraCaptureSession.CaptureCallback, Handler)abortCaptures()  
  
android.hardware.camera2.CameraDevice.StateCallback---onCaptureFailed:   
This method is called instead of onCaptureCompleted(CameraCaptureSession, CaptureRequest, TotalCaptureResult) when the camera device failed to produce a CaptureResult for the request. Other requests are unaffected, and some or all image buffers from the capture may have been pushed to their respective output streams. The default implementation of this method does nothing. See also: capture(CaptureRequest, CameraCaptureSession.CaptureCallback, Handler)captureBurst(List, CameraCaptureSession.CaptureCallback, Handler)setRepeatingRequest(CaptureRequest, CameraCaptureSession.CaptureCallback, Handler)setRepeatingBurst(List, CameraCaptureSession.CaptureCallback, Handler)