Diffusion d’Informations sur Android (BroadCast)

Abdelhak-Djamel Seriai

Qu’est ce que le BroadCast (Diffusion)

- Le BroadCast (diffusion) consiste en la possibilité qu’ont les applications Android pour envoyer ou recevoir des messages du système ou des autres applications Android.
  - Réalisation du patron de conception *« publication-consommation ».*

- Ces diffusions sont envoyées lorsqu'un événement d'intérêt se produit.
  - Par exemple, le système Android envoie des diffusions lorsque divers événements système se produisent tels que le démarrage du système ou changement de mode.

- Les applications peuvent également envoyer des diffusions personnalisées.
  - Par exemple, pour informer d'autres applications de quelque chose qui pourrait les intéresser telle que nouvelles données ont été téléchargées.

Qu’est ce que le BroadCast (Diffusion)

- Les applications peuvent s'inscrire pour recevoir des diffusions spécifiques.
  - Lorsqu'une diffusion est envoyée, le système achemine automatiquement les diffusions vers les applications qui se sont abonnées pour recevoir ce type particulier de diffusion.

- De manière générale, les diffusions peuvent être utilisées comme système de messagerie à travers les applications et en dehors du flux utilisateur normal.
  - Les réponses aux diffusions et d'exécuter des travaux en arrière-plan peuvent contribuer à une baisse des performances du système.

Diffusions « Système »

- Le système envoie automatiquement des diffusions lorsque divers événements du système se produisent
  - Sont envoyées à toutes les applications abonnées pour recevoir l'événement.
    - Par exemple lorsque le système passe en mode avion et hors de celui-ci.

- Le message de diffusion
  - Est enveloppé dans un objet *Intent* dont l'attribut *action* identifie l'événement qui s'est produit.
    - Par exemple *android.intent.action.AIRPLANE_MODE*.
  - L'intention peut également inclure des informations supplémentaires regroupées dans son champ supplémentaire.
    - Par exemple, l'intention du mode avion comprend un supplément booléen qui indique si le mode avion est activé ou non.
- La liste complète des actions de diffusion système est disponible dans le fichier *BROADCAST_ACTIONS.TXT* dans le SDK Android.
  - Chaque action de diffusion est associée à un champ constant.
    - Par exemple, la valeur de la constante *ACTION_AIRPLANE_MODE_CHANGED* est *android.intent.action.AIRPLANE_MODE*.
    - La documentation de chaque action de diffusion est disponible dans son champ constant associé.

1. android.app.action.ACTION_PASSWORD_CHANGED
2. android.app.action.ACTION_PASSWORD_EXPIRING
3. android.app.action.ACTION_PASSWORD_FAILED
4. android.app.action.ACTION_PASSWORD_SUCCEEDED
5. android.app.action.DEVICE_ADMIN_DISABLED
6. android.app.action.DEVICE_ADMIN_DISABLE_REQUESTED
7. android.app.action.DEVICE_ADMIN_ENABLED
8. android.app.action.DEVICE_OWNER_CHANGED
9. android.app.action.INTERRUPTION_FILTER_CHANGED
10. android.app.action.LOCK_TASK_ENTERING
11. android.app.action.LOCK_TASK_EXITING
12. android.app.action.NEXT_ALARM_CLOCK_CHANGED
13. android.app.action.NOTIFICATION_POLICY_ACCESS_GRANTED_CHANGED
14. android.app.action.NOTIFICATION_POLICY_CHANGED
15. android.app.action.PROFILE_PROVISIONING_COMPLETE
16. android.app.action.SYSTEM_UPDATE_POLICY_CHANGED
17. android.bluetooth.a2dp.profile.action.CONNECTION_STATE_CHANGED
18. android.bluetooth.a2dp.profile.action.PLAYING_STATE_CHANGED
19. android.bluetooth.adapter.action.CONNECTION_STATE_CHANGED
20. android.bluetooth.adapter.action.DISCOVERY_FINISHED
21. android.bluetooth.adapter.action.DISCOVERY_STARTED
22. android.bluetooth.adapter.action.LOCAL_NAME_CHANGED
23. android.bluetooth.adapter.action.SCAN_MODE_CHANGED
24. android.bluetooth.adapter.action.STATE_CHANGED
25. android.bluetooth.device.action.ACL_CONNECTED
26. android.bluetooth.device.action.ACL_DISCONNECTED
27. android.bluetooth.device.action.ACL_DISCONNECT_REQUESTED
28. android.bluetooth.device.action.BOND_STATE_CHANGED
29. android.bluetooth.device.action.CLASS_CHANGED
30. android.bluetooth.device.action.FOUND
31. android.bluetooth.device.action.NAME_CHANGED
32. android.bluetooth.device.action.PAIRING_REQUEST
33. android.bluetooth.device.action.UUID
34. android.bluetooth.devicepicker.action.DEVICE_SELECTED
35. android.bluetooth.devicepicker.action.LAUNCH
36. android.bluetooth.headset.action.VENDOR_SPECIFIC_HEADSET_EVENT
37. android.bluetooth.headset.profile.action.AUDIO_STATE_CHANGED
38. android.bluetooth.headset.profile.action.CONNECTION_STATE_CHANGED
39. android.bluetooth.input.profile.action.CONNECTION_STATE_CHANGED
40. android.bluetooth.pan.profile.action.CONNECTION_STATE_CHANGED
41. android.hardware.action.NEW_PICTURE
42. android.hardware.action.NEW_VIDEO
43. android.hardware.hdmi.action.OSD_MESSAGE
44. android.hardware.input.action.QUERY_KEYBOARD_LAYOUTS
45. android.intent.action.ACTION_POWER_CONNECTED
46. android.intent.action.ACTION_POWER_DISCONNECTED
47. android.intent.action.ACTION_SHUTDOWN
48. android.intent.action.AIRPLANE_MODE
49. android.intent.action.APPLICATION_RESTRICTIONS_CHANGED
50. android.intent.action.BATTERY_CHANGED
51. android.intent.action.BATTERY_LOW
52. android.intent.action.BATTERY_OKAY
53. android.intent.action.BOOT_COMPLETED
54. android.intent.action.CAMERA_BUTTON
55. android.intent.action.CONFIGURATION_CHANGED
56. android.intent.action.CONTENT_CHANGED
57. android.intent.action.DATA_SMS_RECEIVED
58. android.intent.action.DATE_CHANGED
59. android.intent.action.DEVICE_STORAGE_LOW
60. android.intent.action.DEVICE_STORAGE_OK

61. android.intent.action.DOCK_EVENT
62. android.intent.action.DOWNLOAD_COMPLETE
63. android.intent.action.DOWNLOAD_NOTIFICATION_CLICKED
64. android.intent.action.DREAMING_STARTED
65. android.intent.action.DREAMING_STOPPED
66. android.intent.action.EXTERNAL_APPLICATIONS_AVAILABLE
67. android.intent.action.EXTERNAL_APPLICATIONS_UNAVAILABLE
68. android.intent.action.FETCH_VOICEMAIL
69. android.intent.action.GTALK_CONNECTED
70. android.intent.action.GTALK_DISCONNECTED
71. android.intent.action.HEADSET_PLUG
72. android.intent.action.HEADSET_PLUG
73. android.intent.action.INPUT_METHOD_CHANGED
74. android.intent.action.LOCALE_CHANGED
75. android.intent.action.MANAGE_PACKAGE_STORAGE
76. android.intent.action.MEDIA_BAD_REMOVAL
77. android.intent.action.MEDIA_BUTTON
78. android.intent.action.MEDIA_CHECKING
79. android.intent.action.MEDIA_EJECT
80. android.intent.action.MEDIA_MOUNTED
81. android.intent.action.MEDIA_NOFS
82. android.intent.action.MEDIA_REMOVED
83. android.intent.action.MEDIA_SCANNER_FINISHED
84. android.intent.action.MEDIA_SCANNER_SCAN_FILE
85. android.intent.action.MEDIA_SCANNER_STARTED
86. android.intent.action.MEDIA_SHARED
87. android.intent.action.MEDIA_UNMOUNTABLE
88. android.intent.action.MEDIA_UNMOUNTED
89. android.intent.action.MY_PACKAGE_REPLACED
90. android.intent.action.NEW_OUTGOING_CALL

91. android.intent.action.NEW_VOICEMAIL
92. android.intent.action.PACKAGE_ADDED
93. android.intent.action.PACKAGE_CHANGED
94. android.intent.action.PACKAGE_DATA_CLEARED
95. android.intent.action.PACKAGE_FIRST_LAUNCH
96. android.intent.action.PACKAGE_FULLY_REMOVED
97. android.intent.action.PACKAGE_INSTALL
98. android.intent.action.PACKAGE_NEEDS_VERIFICATION
99. android.intent.action.PACKAGE_REMOVED
100.android.intent.action.PACKAGE_REPLACED
101.android.intent.action.PACKAGE_RESTARTED
102.android.intent.action.PACKAGE_VERIFIED
103.android.intent.action.PHONE_STATE
104.android.intent.action.PROVIDER_CHANGED
105.android.intent.action.PROXY_CHANGE
106.android.intent.action.REBOOT
107.android.intent.action.SCREEN_OFF
108.android.intent.action.SCREEN_ON
109.android.intent.action.TIMEZONE_CHANGED
110.android.intent.action.TIME_SET
111.android.intent.action.TIME_TICK
112.android.intent.action.UID_REMOVED
113.android.intent.action.USER_PRESENT
114.android.intent.action.WALLPAPER_CHANGED
115.android.media.ACTION_SCO_AUDIO_STATE_UPDATED
116.android.media.AUDIO_BECOMING_NOISY
117.android.media.RINGER_MODE_CHANGED
118.android.media.SCO_AUDIO_STATE_CHANGED
119.android.media.VIBRATE_SETTING_CHANGED
120.android.media.action.CLOSE_AUDIO_EFFECT_CONTROL_SESSION

121.android.media.action.HDMI_AUDIO_PLUG
122.android.media.action.OPEN_AUDIO_EFFECT_CONTROL_SESSION
123.android.net.conn.BACKGROUND_DATA_SETTING_CHANGED
124.android.net.conn.CONNECTIVITY_CHANGE
125.android.net.nsd.STATE_CHANGED
126.android.net.scoring.SCORER_CHANGED
127.android.net.scoring.SCORE_NETWORKS
128.android.net.wifi.NETWORK_IDS_CHANGED
129.android.net.wifi.RSSI_CHANGED
130.android.net.wifi.SCAN_RESULTS
131.android.net.wifi.STATE_CHANGE
132.android.net.wifi.WIFI_STATE_CHANGED
133.android.net.wifi.p2p.CONNECTION_STATE_CHANGE
134.android.net.wifi.p2p.DISCOVERY_STATE_CHANGE
135.android.net.wifi.p2p.PEERS_CHANGED
136.android.net.wifi.p2p.STATE_CHANGED
137.android.net.wifi.p2p.THIS_DEVICE_CHANGED
138.android.net.wifi.supplicant.CONNECTION_CHANGE
139.android.net.wifi.supplicant.STATE_CHANGE
140.android.nfc.action.ADAPTER_STATE_CHANGED
141.android.os.action.DEVICE_IDLE_MODE_CHANGED
142.android.os.action.POWER_SAVE_MODE_CHANGED
143.android.provider.Telephony.SIM_FULL
144.android.provider.Telephony.SMS_CB_RECEIVED
145.android.provider.Telephony.SMS_DELIVER
146.android.provider.Telephony.SMS_EMERGENCY_CB_RECEIVED
147.android.provider.Telephony.SMS_RECEIVED
148.android.provider.Telephony.SMS_REJECTED
149.android.provider.Telephony.SMS_SERVICE_CATEGORY_PROGRAM_DATA_RECEIVED
150.android.provider.Telephony.WAP_PUSH_DELIVER

151.android.provider.Telephony.WAP_PUSH_RECEIVED
152.android.speech.tts.TTS_QUEUE_PROCESSING_COMPLETED
153.android.speech.tts.engine.TTS_DATA_INSTALLED

Réception de Diffusions

- Les applications peuvent recevoir des diffusions de deux manières:
  - Via des récepteurs déclarés par manifeste
    - Si un récepteur de diffusion (*BroadCast Receiver*) est déclaré dans le manifeste, le système lance l'application lorsque la diffusion est envoyée, si l'application n'est pas déjà en cours d'exécution.
  - Via des récepteurs enregistrés dans un contexte.

Récepteurs Déclarés dans le Manifeste

1. Pour déclarer un récepteur de diffusions dans le manifeste :
    1. Spécifiez l'élément *<receiver>* dans le manifeste de l'application.

```
<receiver android:name=".MyBroadcastReceiver"  android:exported="true">
   <intent-filter>
      <action android:name="android.intent.action.BOOT_COMPLETED"/>
      <action android:name="android.intent.action.INPUT_METHOD_CHANGED" />
   </intent-filter>
</receiver>
```

- Les filtres d'intention spécifient les actions de diffusion auxquelles votre récepteur est abonné.

Récepteurs Déclarés dans le Manifeste

2. Définir une Sous-classe de *BroadcastReceiver* et implémentez *onReceive (Context, Intent)*.
   - Le récepteur de diffusion dans l'exemple suivant enregistre et affiche le contenu de la diffusion:

```
public class MyBroadcastReceiver extends BroadcastReceiver {
        private static final String TAG = "MyBroadcastReceiver";
        @Override
        public void onReceive(Context context, Intent intent) {
            StringBuilder sb = new StringBuilder();
            sb.append("Action: " + intent.getAction() + "\n");
            sb.append("URI: " + intent.toUri(Intent.URI_INTENT_SCHEME).toString() + "\n");
            String log = sb.toString();
            Log.d(TAG, log);
            Toast.makeText(context, log, Toast.LENGTH_LONG).show();
        }
    }
```

Récepteurs Déclarés dans le Manifeste

- Le gestionnaire de package système enregistre le récepteur lorsque l'application est installée.
  - Le récepteur devient alors un point d'entrée distinct dans l'application:
    - Ce qui signifie que le système peut démarrer l'application et diffuser la diffusion si l'application n'est pas en cours d'exécution.

- Le système crée un nouvel objet composant *BroadcastReceiver* pour gérer chaque diffusion :
  - Cet objet n'est valide que pendant la durée de l'appel à *onReceive (Context, Intent).*
  - Une fois que le code de l'application sort de cette méthode, le système considère que ce composant n'est plus actif.

Récepteurs Déclarés dans le Contexte

1. Création d'une instance de *BroadcastReceiver*.

```
BroadcastReceiver br = new MyBroadcastReceiver();
```

2. Création d'un *IntentFilter* et enregistrer le récepteur en appelant *registerReceiver* (*BroadcastReceiver, IntentFilter*):

```
IntentFilter filter = new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);
   filter.addAction(Intent.ACTION_AIRPLANE_MODE_CHANGED);
   this.registerReceiver(br, filter);
```

- Pour s'inscrire aux diffusions locales, appeler plutôt *LocalBroadcastManager.registerReceiver (BroadcastReceiver, IntentFilter)*.

- Les récepteurs enregistrés dans le contexte reçoivent des émissions tant que leur contexte d'enregistrement est valide.
  - Par exemple
    - Si l'inscription est faite dans un contexte d'activité, les diffusions sont reçues tant que l'activité n'est pas détruite.
    - Si l'inscription est faite dans le contexte de l'application, les diffusions sont reçues tant que l'application est en cours d'exécution.

Récepteurs Déclarés dans le Contexte

3. Pour arrêter de recevoir des diffusions, appeler *unregisterReceiver (android.content.BroadcastReceiver)*.

- Endroit où enregistrer et de se désenregistrer le récepteur,
  - Récepteur enregistré dans onCreate (Bundle) en utilisant le contexte de l'activité :
    - Le désinscrire dans *onDestroy ()* pour éviter de perdre le récepteur hors du contexte d'activité.
  - Récepteur enregistré dans *onResume ()* :
    - Le désinscrire dans *onPause ()*.
  - Ne pas désinscrire dans *onSaveInstanceState (Bundle)*
    - Cela n'est pas appelé si l'utilisateur revient dans la pile d'historique.

Effets sur l’Etat du Processus

- L'état du *BroadcastReceiver* (en cours d'exécution ou non) affecte l'état de son processus de confinement
  - Peut affecter à son tour sa probabilité d'être tué par le système.
    - Par exemple,
      - Lorsqu'un processus exécute un récepteur; c'est-à-dire qu'il exécute actuellement le code dans sa méthode *onReceive (),* il est considéré comme un processus de premier plan.
        - Le système maintient le processus en cours sauf en cas de pression de mémoire extrême.
      - Cependant, une fois que le code sort de *onReceive (),* le *BroadcastReceiver* n'est plus actif.
  - Le processus hôte du récepteur devient aussi important que les autres composants d'application qui y sont exécutés.

Effets sur l'Etat du Processus

- Si ce processus héberge uniquement un récepteur déclaré dans le manifeste :
  - A la sortie de *onReceive ()*, le système considère que son processus est un processus de faible priorité et peut le tuer pour rendre les ressources disponibles pour d'autres processus plus importants.
    - Pour cette raison, ne pas démarrer de longs threads d'arrière-plan à partir d'un récepteur de diffusion.
    - Méthodes efficaces de traitement de réception de diffusions :
      - Soit appeler *goAsync ()* : pour traiter la diffusion dans un thread d'arrière-plan.
      - Ou planifier un *JobService* à partir du récepteur à l'aide de *JobScheduler*, afin que le système sache que le processus continue d'être actif

Effets sur l'Etat du Processus

- Exemple :
  - Un *BroadcastReceiver* qui utilise *goAsync ()* pour signaler qu'il a besoin de plus de temps pour terminer une fois *onReceive ()* terminé.
  - Utile si le travail à terminer dans onReceive () est suffisamment long pour être traité dans le thread d'interface utilisateur (> 16 ms), ce qui le rend mieux adapté à un thread d'arrière-plan.

```
public class MyBroadcastReceiver extends BroadcastReceiver {
    private static final String TAG = "MyBroadcastReceiver";

    @Override
    public void onReceive(Context context, Intent intent) {
        final PendingResult pendingResult = goAsync();
        Task asyncTask = new Task(pendingResult, intent);
        asyncTask.execute();
    }

    private static class Task extends AsyncTask<String, Integer, String> {

        private final PendingResult pendingResult;
        private final Intent intent;

        private Task(PendingResult pendingResult, Intent intent) {
            this.pendingResult = pendingResult;
            this.intent = intent;
        }
```

Effets sur l'Etat du Processus

```
        @Override
        protected String doInBackground(String... strings) {
            StringBuilder sb = new StringBuilder();
            sb.append("Action: " + intent.getAction() + "\n");
            sb.append("URI: " + intent.toUri(Intent.URI_INTENT_SCHEME).toString() + "\n");
            String log = sb.toString();
            Log.d(TAG, log);
            return log;
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            // Must call finish() so the BroadcastReceiver can be recycled.
            pendingResult.finish();
        }
    }
}
```

Envoi de Diffusions

- Android offre trois façons pour les applications d'envoyer une diffusion:
  1. La méthode *sendOrderedBroadcast (Intent, String)*
     - Envoie des diffusions à un récepteur à la fois.
     - Au fur et à mesure que chaque récepteur s'exécute à son tour, il peut propager un résultat au récepteur suivant, ou il peut annuler complètement la diffusion afin qu'elle ne soit pas transmise à d'autres récepteurs.
     - Les récepteurs de commande exécutés peuvent être contrôlés avec l'attribut *android:priority* du filtre d'intention correspondant;
       - Les récepteurs ayant la même priorité seront exécutés dans un ordre arbitraire.

  2. La méthode *sendBroadcast (Intent)* → C'est ce qu'on appelle une diffusion normale.
     - Envoie des diffusions à tous les récepteurs dans un ordre non défini.
     - Ceci est plus efficace, mais signifie que les récepteurs ne peuvent pas lire les résultats des autres récepteurs, propager les données reçues de la diffusion ou interrompre la diffusion.

  3. La méthode *LocalBroadcastManager.sendBroadcast*
     - Envoie des diffusions aux récepteurs qui se trouvent dans la même application que l'expéditeur.
     - Remarque : Si pas besoin d'envoyer des diffusions à travers les applications, utilisation des diffusions locales.
       - L'implémentation est beaucoup plus efficace (aucune communication interprocessus nécessaire) et pas besoin de se soucier des problèmes de sécurité liés à la possibilité pour d'autres applications de recevoir ou d'envoyer des diffusions.

Envoi de Diffusions

- Exemple : Envoyer une diffusion en créant une intention et en appelant *sendBroadcast (intention).*

```
Intent intent = new Intent();
intent.setAction("com.example.broadcast.MY_NOTIFICATION");
intent.putExtra("data","Notice me senpai!");
sendBroadcast(intent);
```

  - Le message de diffusion est encapsulé dans un objet *Intent*.
  - La chaîne/attribut d'action de l'intention doit fournir la syntaxe du nom du package Java de l'application et identifier de manière unique l'événement de diffusion.
  - Possibilité d'attacher des informations supplémentaires à l'intention avec *putExtra (String, Bundle).*
  - Possibilité également de limiter une diffusion à un ensemble d'applications dans le même organisation en appelant *setPackage (String)*.

- Remarque: Les intentions sont utilisées à la fois pour l'envoi de diffusions et le démarrage d'activités avec *startActivity (Intent)*
  - Mais les deux actions sont totalement indépendantes :
    - Les récepteurs de diffusion ne peuvent ni voir ni capturer les intentions utilisées pour démarrer une activité; de même, lorsque une intention est diffusée, il n'est pas possible de trouver ou démarrer une activité.

Restreindre les Diffusions avec des Autorisations

- Les autorisations permettent de :
  - Restreindre les diffusions à l'ensemble d'applications qui détiennent certaines autorisations.
  - Appliquer des restrictions à l'expéditeur ou au destinataire d'une diffusion.

- Envoi avec autorisations :
  - Il est possible de spécifier un paramètre d'autorisation lorsque on appelle
    - *sendBroadcast (Intent, String)*
    - Ou *sendOrderedBroadcast (Intent, String, BroadcastReceiver, Handler, int, String, Bundle)*

  - Seuls les récepteurs qui ont demandé cette autorisation avec la balise spécifique dans leur manifeste (et ont ensuite obtenu l'autorisation) peuvent recevoir la diffusion.

  - Les autorisations personnalisées sont enregistrées lors de l'installation de l'application.
    - Donc, l'application qui définit l'autorisation personnalisée doit être installée avant l'application qui l'utilise.

Restreindre les Diffusions avec des Autorisations

- Envoi avec autorisations :
  - Exemple : Envoie une diffusion

```
sendBroadcast(new Intent("com.example.NOTIFY"),
Manifest.permission.SEND_SMS);
```

  - Pour recevoir la diffusion, l'application réceptrice doit demander l'autorisation comme indiqué ci-dessous:

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

Restreindre les Diffusions avec des Autorisations

- Réception avec autorisations
  1. Spécification d'un paramètre d'autorisation lors de l'enregistrement d'un récepteur de diffusion :
     - Avec *registerReceiver (BroadcastReceiver, IntentFilter, String, Handler).*
     - ou dans la balise *<receiver>* du manifeste.

     - Seuls les diffuseurs qui ont demandé la permission avec la *<uses-permission>* tag dans leur manifeste (et par la suite obtenu l'autorisation si elle est dangereuse) peut envoyer une intention au récepteur.

     - Par exemple, supposons que l'application de réception dispose d'un récepteur déclaré manifeste, comme indiqué ci-dessous:

```
<receiver android:name=".MyBroadcastReceiver"
        android:permission="android.permission.SEND_SMS">
   <intent-filter>
      <action android:name="android.intent.action.AIRPLANE_MODE"/>
   </intent-filter>
</receiver>
```

Restreindre les Diffusions avec des Autorisations

- Réception avec autorisations
  2. Application de réception qui dispose d'un récepteur enregistré dans le contexte,

```
IntentFilter filter = new IntentFilter(Intent.ACTION_AIRPLANE_MODE_CHANGED);
registerReceiver(receiver, filter, Manifest.permission.SEND_SMS, null );
```

  - Ensuite, pour pouvoir envoyer des diffusions à ces récepteurs, l'application d'envoi doit demander l'autorisation comme indiqué ci-dessous:

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

Considérations de Sécurité et Meilleures Pratiques

1. Si pas besoin d'envoyer des diffusions à des composants en dehors de l'application, envoyer et recever des diffusions locales avec *LocalBroadcastManager* qui est disponible dans la bibliothèque de support.
   - Le LocalBroadcastManager est beaucoup plus efficace (aucune communication interprocessus nécessaire)

2. Si de nombreuses applications se sont inscrites pour recevoir la même diffusion dans leur manifeste, le système peut lancer de nombreuses applications, ce qui a un impact substantiel sur les performances de l'appareil et sur l'expérience utilisateur.
   - Pour éviter cela, à préférer l'utilisation de l'enregistrement de contexte plutôt que la déclaration de manifeste.
   - Parfois, le système Android lui-même impose l'utilisation de récepteurs enregistrés dans le contexte. Par exemple, la diffusion *CONNECTIVITY_ACTION* est délivrée uniquement aux récepteurs enregistrés dans le contexte.

Considérations de Sécurité et Meilleures Pratiques

3. Ne pas diffuser d'informations sensibles à l'aide d'une intention implicite.
   - Les informations peuvent être lues par n'importe quelle application qui s'inscrit pour recevoir la diffusion.
   - Il existe trois façons de contrôler qui peut recevoir les diffusions:
     - Spécifier une autorisation lors de l'envoi d'une diffusion.
     - Dans Android 4.0 et supérieur, spécifier un package avec *setPackage (String)* lors de l'envoi d'une diffusion. Le système limite la diffusion à l'ensemble des applications qui correspondent au package.
     - Envoyer des diffusions locales avec *LocalBroadcastManager*.

- Lorsque un récepteur est enregistré, n'importe quelle application peut envoyer des diffusions potentiellement malveillantes au récepteur d'une autre application.
  - Il existe trois façons de limiter les diffusions reçues par une application:
    - Spécifier une autorisation lors de l'enregistrement d'un récepteur de diffusion.
    - Pour les récepteurs déclarés par manifeste, il est possible de définir l'attribut *android: exports* sur "false" dans le manifeste. Le récepteur ne reçoit pas de diffusion de sources extérieures à l'application.
    - Se limiter aux diffusions locales uniquement avec *LocalBroadcastManager*.

- L'espace de noms pour les actions de diffusion est global. Il faut s'assurer que les noms d'actions et autres chaînes sont écrits dans un espace de noms que vous possédez, sinon vous risquez d'entrer en conflit par inadvertance avec d'autres applications.

Considérations de Sécurité et Meilleures Pratiques

- Étant donné que la méthode *onReceive (Context, Intent)* d'un récepteur s'exécute sur le thread principal, elle doit s'exécuter et sortir rapidement.
  - Si besoin d'effectuer un travail de longue durée, faire attention à la génération de threads ou au démarrage de services d'arrière-plan, car le système peut tuer l'intégralité du processus après le retour de *onReceive ().*

- Ne démarrez pas d'activités à partir de récepteurs de diffusion car l'expérience utilisateur est discordante; surtout s'il y a plus d'un récepteur. Envisagez plutôt d'afficher une notification.

Modifications des Diffusions « Système »

- Le comportement du système de diffusion a évolué avec les différentes versions d'Android:
  - Android 9
    - Exemple : À partir d'Android 9 (API niveau 28), la diffusion *NETWORK_STATE_CHANGED_ACTION* ne reçoit pas d'informations sur l'emplacement de l'utilisateur ni sur des données personnellement identifiables.
  - Android 8.0
    - Exemple : À partir d'Android 8.0 (API niveau 26), le système impose des restrictions supplémentaires aux récepteurs déclarés dans le manifeste.
      - Si l'application cible Android 8.0 ou version ultérieure, le manifeste ne peut pas être utilisé pour déclarer un récepteur pour la plupart des diffusions implicites (diffusions qui ne ciblent pas spécifiquement votre application).
  - Android 7.0
    - Android 7.0 (API niveau 24) et supérieur n'envoie pas les diffusions système suivantes:
      - *ACTION_NEW_PICTURE*
      - *ACTION_NEW_VIDEO*
    - Les applications doivent enregistrer la diffusion *CONNECTIVITY_ACTION* à l'aide de *registerReceiver* (*BroadcastReceiver, IntentFilter*).
    - La déclaration d'un récepteur dans le manifeste ne fonctionne pas.