<receiver>

SYNTAX:

```
<receiver android:directBootAware=["true" | "false"]
          android:enabled=["true" | "false"]
          android:exported=["true" | "false"]
          android:icon="drawable resource"
          android:label="string resource"
          android:name="string"
          android:permission="string"
          android:process="string" >
    . . .
</receiver>
```

CONTAINED IN:

`<application>`

CAN CONTAIN:

`<intent-filter>`
`<meta-data>`

DESCRIPTION:

Declares a broadcast receiver (a `BroadcastReceiver` subclass) as one of the application's components. Broadcast receivers enable applications to receive intents that are broadcast by the system or by other applications, even when other components of the application are not running.

There are two ways to make a broadcast receiver known to the system: One is declare it in the manifest file with this element. The other is to create the receiver dynamically in code and register it with the `Context.registerReceiver()` method. For more information about how to dynamically create receivers, see the `BroadcastReceiver` class description.

Warning: Limit how many broadcast receivers you set in your app. Having too many broadcast receivers can affect your app's performance and the battery life of users' devices. For more information about APIs you can use instead of the `BroadcastReceiver` class for scheduling background work, see Background Optimizations.

ATTRIBUTES:

`android:directBootAware`

Whether or not the broadcast receiver is *direct-boot aware*; that is, whether or not it can run before the user unlocks the device.

Note: During Direct Boot, a broadcast receiver in your application can only access the data that is stored in *device protected* storage.

The default value is `"false"`.

`android:enabled`

Whether or not the broadcast receiver can be instantiated by the system — "`true`" if it can be, and "`false`" if not. The default value is "`true`".

The `<application>` element has its own `enabled` attribute that applies to all application components, including broadcast receivers. The `<application>` and `<receiver>` attributes must both be "`true`" for the broadcast receiver to be enabled. If either is "`false`", it is disabled; it cannot be instantiated.

`android:exported`

Whether or not the broadcast receiver can receive messages from sources outside its application — "`true`" if it can, and "`false`" if not. If "`false`", the only messages the broadcast receiver can receive are those sent by components of the same application or applications with the same user ID.

The default value depends on whether the broadcast receiver contains intent filters. The absence of any filters means that it can be invoked only by Intent objects that specify its exact class name. This implies that the receiver is intended only for application-internal use (since others would not normally know the class name). So in this case, the default value is "`false`". On the other hand, the presence of at least one filter implies that the broadcast receiver is intended to receive intents broadcast by the system or other applications, so the default value is "`true`".

This attribute is not the only way to limit a broadcast receiver's external exposure. You can also use a permission to limit the external entities that can send it messages (see the `permission` attribute).

`android:icon`

An icon representing the broadcast receiver. This attribute must be set as a reference to a drawable resource containing the image definition. If it is not set, the icon specified for the application as a whole is used instead (see the `<application>` element's `icon` attribute).

The broadcast receiver's icon — whether set here or by the `<application>` element — is also the default icon for all the receiver's intent filters (see the `<intent-filter>` element's `icon` attribute).

`android:label`

A user-readable label for the broadcast receiver. If this attribute is not set, the label set for the application as a whole is used instead (see the `<application>` element's `label` attribute).

The broadcast receiver's label — whether set here or by the `<application>` element — is also the default label for all the receiver's intent filters (see the `<intent-filter>` element's `label` attribute).

The label should be set as a reference to a string resource, so that it can be localized like other strings in the user interface. However, as a convenience while you're developing the application, it can also be set as a raw string.

`android:name`

The name of the class that implements the broadcast receiver, a subclass of `BroadcastReceiver`. This should be a fully qualified class name (such as, "`com.example.project.ReportReceiver`"). However, as a shorthand, if the first character of the name is a period (for example, "`. ReportReceiver`"), it is appended to the package name specified in the `<manifest>` element.

Once you publish your application, you should not change this name (unless you've set `android:exported="false"`).

There is no default. The name must be specified.

`android:permission`

The name of a permission that broadcasters must have to send a message to the broadcast receiver. If this attribute is not set, the permission set by the `<application>` element's `permission` attribute applies to the broadcast receiver. If neither attribute is set, the receiver is not protected by a permission.

For more information on permissions, see the Permissions section in the introduction and a separate document, Security and Permissions.

`android:process`

The name of the process in which the broadcast receiver should run. Normally, all components of an application run in the default process created for the application. It has the same name as the application package. The `<application>` element's `process` attribute can set a different default for all components. But each component can override the default with its own `process` attribute, allowing you to spread your application across multiple processes.

If the name assigned to this attribute begins with a colon (':'), a new process, private to the application, is created when it's needed and the broadcast receiver runs in that process. If the process name begins with a lowercase character, the receiver will run in a global process of that name, provided that it has permission to do so. This allows components in different applications to share a process, reducing resource usage.

INTRODUCED IN:

API Level 1