<provider>

SYNTAX:

```
<provider android:authorities="list"
          android:directBootAware=["true" | "false"]
          android:enabled=["true" | "false"]
          android:exported=["true" | "false"]
          android:grantUriPermissions=["true" | "false"]
          android:icon="drawable resource"
          android:initOrder="integer"
          android:label="string resource"
          android:multiprocess=["true" | "false"]
          android:name="string"
          android:permission="string"
          android:process="string"
          android:readPermission="string"
          android:syncable=["true" | "false"]
          android:writePermission="string" >
    . . .
</provider>
```

CONTAINED IN:

`<application>`

CAN CONTAIN:

`<meta-data>`
`<grant-uri-permission>`
`<path-permission>`

DESCRIPTION:

Declares a content provider component. A content provider is a subclass of `ContentProvider` that supplies structured access to data managed by the application. All content providers in your application must be defined in a `<provider>` element in the manifest file; otherwise, the system is unaware of them and doesn't run them.

You only declare content providers that are part of your application. Content providers in other applications that you use in your application should not be declared.

The Android system stores references to content providers according to an **authority** string, part of the provider's **content URI**. For example, suppose you want to access a content provider that stores information about health care professionals. To do this, you call the method `ContentResolver.query()`, which among other arguments takes a URI that identifies the provider:

```
content://com.example.project.healthcareprovider/nurses/rn
```

The `content:` **scheme** identifies the URI as a content URI pointing to an Android content provider. The authority `com.example.project.healthcareprovider` identifies the provider itself; the Android system looks up the authority in its list of known providers and their authorities. The substring `nurses/rn` is a **path**, which the content provider can use to identify subsets of the provider data.

Notice that when you define your provider in the `<provider>` element, you don't include the scheme or the path in the `android:name` argument, only the authority.

For information on using and developing content providers, see the API Guide, Content Providers.

ATTRIBUTES:

`android:authorities`

A list of one or more URI authorities that identify data offered by the content provider. Multiple authorities are listed by separating their names with a semicolon. To avoid conflicts, authority names should use a Java-style naming convention (such as `com.example.provider.cartoonprovider`). Typically, it's the name of the `ContentProvider` subclass that implements the provider

There is no default. At least one authority must be specified.

`android:enabled`

Whether or not the content provider 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 content providers. The `<application>` and `<provider>` attributes must both be "`true`" (as they both are by default) for the content provider to be enabled. If either is "`false`", the provider is disabled; it cannot be instantiated.

`android:directBootAware`

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

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

The default value is `"false"`.

`android:exported`

Whether the content provider is available for other applications to use:

- `true`: The provider is available to other applications. Any application can use the provider's content URI to access it, subject to the permissions specified for the provider.
- `false`: The provider is not available to other applications. Set `android:exported="false"` to limit access to the provider to your applications. Only applications that have the same user ID (UID) as the provider will have access to it.

Because this attribute was introduced in API level 17, all devices running API level 16 and lower behave as though this attribute is set `"true"`. If you set `android:targetSdkVersion` to 17 or higher, then the default value is `"false"` for devices running API level 17 and higher.

You can set `android:exported="false"` and still limit access to your provider by setting permissions with the `permission` attribute.

`android:grantUriPermissions`

Whether or not those who ordinarily would not have permission to access the content provider's data can be granted permission to do so, temporarily overcoming the restriction imposed by the `readPermission`, `writePermission`, and `permission` attributes — "`true`" if permission can be granted, and "`false`" if not. If "`true`", permission can be granted to any of the content provider's data. If "`false`", permission can be granted only to the data subsets listed in `<grant-uri-permission>` subelements, if any. The default value is "`false`".

Granting permission is a way of giving an application component one-time access to data protected by a permission. For example, when an e-mail message contains an attachment, the mail application may call upon the appropriate viewer to open it, even though the viewer doesn't have general permission to look at all the content provider's data.

In such cases, permission is granted by `FLAG_GRANT_READ_URI_PERMISSION` and `FLAG_GRANT_WRITE_URI_PERMISSION` flags

in the Intent object that activates the component. For example, the mail application might put `FLAG_GRANT_READ_URI_PERMISSION` in the Intent passed to `Context.startActivity()`. The permission is specific to the URI in the Intent.

If you enable this feature, either by setting this attribute to "`true`" or by defining `<grant-uri-permission>` subelements, you must call `Context.revokeUriPermission()` when a covered URI is deleted from the provider.

See also the `<grant-uri-permission>` element.

`android:icon`

An icon representing the content provider. 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).

`android:initOrder`

The order in which the content provider should be instantiated, relative to other content providers hosted by the same process. When there are dependencies among content providers, setting this attribute for each of them ensures that they are created in the order required by those dependencies. The value is a simple integer, with higher numbers being initialized first.

`android:label`

A user-readable label for the content provided. 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 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:multiprocess`

If the app runs in multiple processes, this attribute determines whether multiple instances of the content provder are created. If `true`, each of the app's processes has its own content provider object. If `false`, the app's processes share only one content provider object. The default value is `false`.

Setting this flag to `true` may improve performance by reducing the overhead of interprocess communication, but it also increases the memory footprint of each process.

`android:name`

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

There is no default. The name must be specified.

`android:permission`

The name of a permission that clients must have to read or write the content provider's data. This attribute is a convenient way of setting a single permission for both reading and writing. However, the `readPermission` and `writePermission` attributes take precedence over this one. If the `readPermission` attribute is also set, it controls access for querying the content provider. And if the `writePermission` attribute is set, it controls access for modifying the provider's data.

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 content provider 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 activity runs in that process. If the process name begins with a lowercase character, the activity 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.

`android:readPermission`

A permission that clients must have to query the content provider. See also the `permission` and `writePermission` attributes.

`android:syncable`

Whether or not the data under the content provider's control is to be synchronized with data on a server — "`true`" if it is to be synchronized, and "`false`" if not.

`android:writePermission`

A permission that clients must have to make changes to the data controlled by the content provider. See also the `permission` and `readPermission` attributes.

INTRODUCED IN:

API Level 1

SEE ALSO:

Content Providers