Appendix

Device File system

You can use the BlackBerry 10 device's file system in order to store your application's data and share files with other apps running on the device. This appendix gives you an overview of your application's home directory's structure and the corresponding directory permissions.

File system structure

When you deploy a Cascades app on a device, an application working directory or *sandbox* is created for your app by the BlackBerry 10 runtime (see Figure A-1).

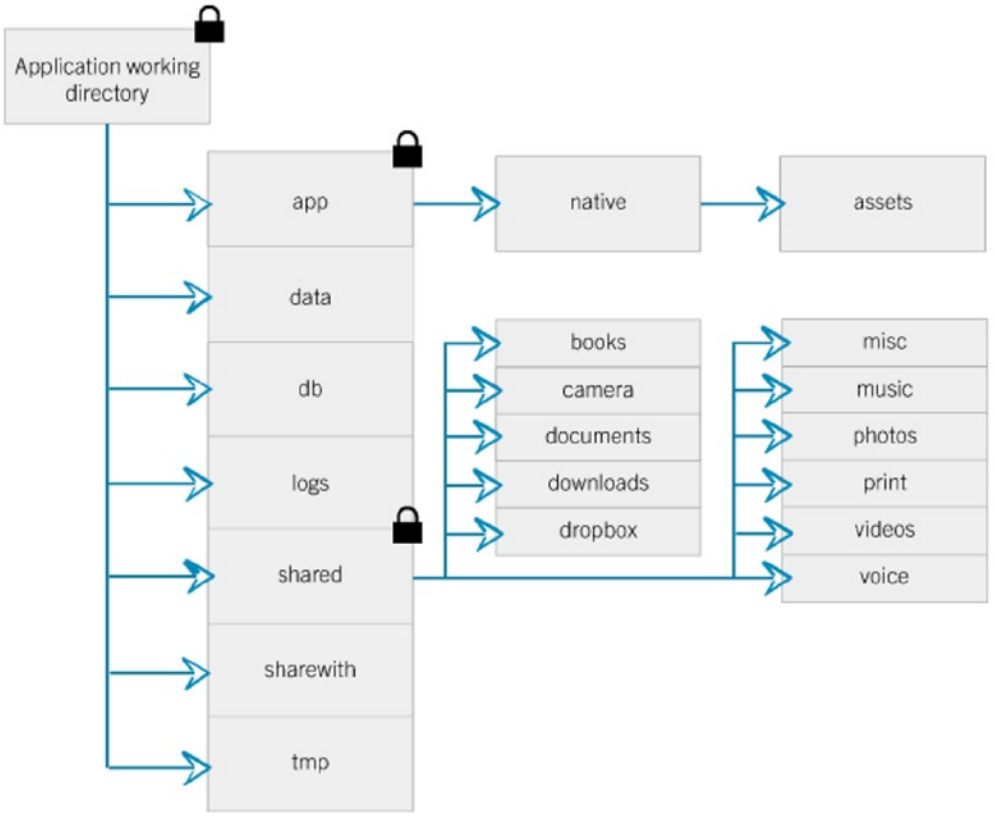

Figure A-1. *Application Sandbox (image source: BlackBerry)*

The following list gives you a description of the directory structure created for your app by the BlackBerry 10 runtime:

- The application's working directory, or sandbox, is the directory where your application is started. When you deploy your app on the device, additional sub-directories are also created for you by the BlackBerry 10 runtime. Depending on the sub-directory, your app has either read-only or read-write access.
- `app`: The app directory is where your application's binaries and resources such as QML files are deployed. You have read-only access to this directory (your app's QML files are located under `app/native/assets`).
- `data`: The data directory is where you can store your application's data, including new data created by your app. Your app has read-write access to this directory. Your app can also create new sub-directories in order to organize its data. Note that when the user removes your app, the contents of this directory will be also removed.
- `db`: You can store database files in this directory (note that the directory will not always be created for you automatically). In practice, and for convenience reasons, you can simply store your database files in the data directory instead.

This is a read-only directory (in other words even if you store database files here, you can't update their contents).

- `logs`: Your application's logs, including the `stderr` and `stdout`, are written in this directory. Your app has read-write access to the directory.
- `shared`: This directory will be created for you if your app has the `access_shared` permission specified in its bar-descriptor.xml file (the directory is in fact a link to the `/accounts/1000/shared` folder on your device). This is a read-only directory; however your app can write in its sub-directories. The different sub-directories correspond to locations where other apps share their own files (for example, `shared/camera` is where the Camera app stores images taken with the device's camera). Note that when the user removes your app from the device, the files stored in this folder's sub-directories by your app are not removed.
- `sharewith`: Contains files that your app can pass to other apps using the Invocation Framework (see Chapter 10).
- `tmp`: This is a folder where your app can store temporary data. The BlackBerry 10 OS can delete the contents of this folder without notification when your app is closed.

You can use the `QDir` class in C++ in order to update your application's sandbox directories (of course your app must have read-write permissions on the updated directories. Also note that a default-constructed `QDir` instance will always point to the app's working directory).

- `QString QDir::currentPath()`: Returns the application's sandbox absolute path. This is a static method that you can also use in order to determine the absolute path of directories located under the application sandbox (for example, to get the absolute path of the data directory, you can use the following method call: `QString dataPath = QDir::currentPath()+"/data"`. You can also use this method in order to build URLs pointing to resources located on the file system. For example: `QUrl("file://"+QDir::currentPath()+"/shared/camera/file001.jpg");`
- `bool QDir::mkpath(conts QString& path)`: Creates the directory path located under the app's working directory. All parent directories required to create the child directory will also be created if necessary. Returns true if successful.

Finally, to access files in the app sandbox, you can use the C++ `QFile` class. For example, this can be very useful if you want to download or upload files using the Qt networking classes (see Chapter 7; note that you can always pass a `QFile` as a second parameter to the `QNetworkAccessManager::post()` method).

> **Note** You can find additional information about QFile and QDir at the following URLs:
>
> http://developer.blackberry.com/native/reference/cascades/qfile.html
>
> http://developer.blackberry.com/native/reference/cascades/qdir.html