io-fs-media Reference

QNX Software Systems Limited
1001 Farrar Road
Ottawa, Ontario
K2K 0B3
Canada

Voice: +1 613 591-0931
Fax: +1 613 591-3579
Email: info@qnx.com
Web: http://www.qnx.com/

Electronic edition published: April 13, 2015

Typographical conventions

Throughout this manual, we use certain typographical conventions to distinguish technical terms. In general, the conventions we use conform to those found in IEEE POSIX publications.

The following table summarizes our conventions:

Reference	Example
Code examples	`if( stream == NULL)`
Command options	`-lR`
Commands	`make`
Constants	NULL
Data types	**unsigned short**
Environment variables	*PATH*
File and pathnames	**/dev/null**
Function names	*exit()*
Keyboard chords	**Ctrl–Alt–Delete**
Keyboard input	`Username`
Keyboard keys	**Enter**
Program output	`login:`
Variable names	*stdin*
Parameters	*parm1*
User-interface components	**Navigator**
Window title	**Options**

We use an arrow in directions for accessing menu items, like this:

You'll find the Other... menu item under **Perspective** → **Show View**.

We use notes, cautions, and warnings to highlight important messages:

Notes point out something important or useful.

CAUTION: Cautions tell you about commands or procedures that may have unwanted or undesirable side effects.

WARNING: Warnings tell you about commands or procedures that could be dangerous to your files, your hardware, or even yourself.

Note to Windows users

In our documentation, we typically use a forward slash (/) as a delimiter in pathnames, including those pointing to Windows files. We also generally follow POSIX/UNIX filesystem conventions.

Technical support

Technical assistance is available for all supported products.

To obtain technical support for any QNX product, visit the Support area on our website (*www.qnx.com*). You'll find a wide range of support options, including community forums.

Chapter 1
io-fs-media Reference

Media filesystem interface

Synopsis:

```
io-fs-media [-a] [-b] [-cTYPE=num] [-ddriver,options] [-t] [-v]
```

Options:

-a

Start all built-in drivers. In this release, only tmp is built in.

-b

Don't run in the background. This setting is used for debugging.

-cTYPE=*num*

Set a cache size. The type of cache is specified by `TYPE` and can be one of:

- attrs — number of attributes (default: 32)
- bundles — number of 64 KB bundles (default: 2)
- files — maximum number of files per filesystem (default: 1048576)
- maps — number of extent mappings (default: 512)
- mcache — maximum size of the meta cache (default: 16 KB)
- meta — size of the metadata cache (default: 16 KB)
- pages — number of 4 KB clusters (default: 256)
- throngs — number of 1 MB throngs (default: 0)
- wads — number of 256 KB wads (default: 0)

You can specify multiple `-c` options on the command line.

-d*driver*,*options*

Start the specified driver with the specified options. The available drivers are:

- tmp (built into **io-fs-media**)
- MTP (**iofs-pfs.so**)
- iPod (**iofs-ipod.so**)

You should start each driver in its own process because **io-fs-media** is single-threaded.

-t

Print the built-in tables to **stdout**, then exit.

`-v`

Enable the verbose option for all mounted filesystems.

Description:

The **io-fs-media** filesystem interface makes distinct media devices and filesystems appear as POSIX-compliant filesystems under QNX Neutrino.

The **io-fs-media** service supports these media devices and filesystems:

RAM filesystem (tmp)

The tmp driver is statically linked to the **io-fs-media** binary and supports a temporary POSIX filesystem in RAM. For information on configuring this driver, see "*Starting the RAM filesystem:* (p. 8)".

MTP filesystem

The MTP driver supports access to files on MTP devices. This functionality is provided by the **iofs-pfs.so** library, which is loaded at runtime. A USB stack is required to use this driver. For information about starting and configuring the MTP driver, see "*iofs-pfs.so Reference* (p. 10)".

iPod filesystem

The iPod driver supports access to files on iPod devices. This functionality is provided by the **iofs-ipod.so** library, which is loaded at runtime. For information about starting and configuring this driver, see the iPod documentation shipped in the iPod download package.

Starting the RAM filesystem:

The tmp driver creates, in RAM, a temporary POSIX filesytem that exists for the duration of the **io-fs-media** process. You can configure this temporary filesystem by issuing **io-fs-media** commands with the `-dtmp` option followed by a comma-separated list defining one or more tmp driver options. The driver supports these options:

`use`

Print a usage message and then exit.

`maxsize=`*size*

The maximum size for the filesystem (in MB). You must provide a number followed by `M` to indicate the unit (e.g., `maxsize=12M`).

`minsize=`*size*

The minimum size for the filesystem (in MB). You must provide a number followed by `M` to indicate the unit (e.g., `minsize=3M`).

`mount=`*path*

Assign a mountpoint for the RAM filesystem. The default mountpoint is **/fs/tmpfs**.

noglob

Don't store the filesystem in global memory. This option is applicable on ARM targets only.

Example:

Suppose you issue this command to configure a temporary filesystem:

```
# io-fs-media -d tmp,minsize=3M,maxsize=12M -cpages=4 -cbundles=0
```

The `minsize` option sets a minimum filesystem size of 3 MB, which is allocated at startup as contiguous memory. The memory allocation can increase up to the size specified by `maxsize` as needed. Memory allocated after startup is not necessarily contiguous.

With ARM targets, the behavior is different: at startup, memory is allocated contiguously up to the size in `maxsize`.

iofs-pfs.so Reference

MTP driver for io-fs-media

Synopsis:

```
io-fs-media -dpfs, options
```

Options:

albdir

Treat Abstract Audio Album (**.alb**) objects, which contain references to tracks from an album, as directories. By default, **iofs-pfs.so** treats these objects as files, in which case they can't be used to directly access album tracks (because the track locations are device-dependent).

If it's faster to perform a directed synchronization of the **Albums** directory than to synchronize the entire mediastore, you should use the `albdir` option. For information on directed synchronizations, see “Full, directed, and file synchronizations” in the *Multimedia Synchronizer Developer's Guide*.

.alb objects as files

With this default driver setting, synchronization operations won't find tracks through **.alb** objects, preventing duplicate database entries when the entire mediastore is synchronized. However, you can't browse the contents of **.alb** objects by performing a directed synchronization of the **Albums** directory.

.alb objects as directories

When the `albdir` option is used, synchronization operations might synchronize the contents of **.alb** objects from multiple directories (e.g., **Albums** and **Music**) and insert duplicate entries into the database for each track referenced in multiple **.alb** objects. However, you can browse the contents of these objects through directed synchronizations.

debug=*n*

Set the debug level, where *n* is between `0` and `5`. A value of `0` disables debug messaging while `5` generates maximum output. The debugging messages are logged to **sloginfo**.

device=*bus_no* : *device_no* : *interface_no*

Assign numbers for the bus, device, and interface to use for an MTP device. This option allows **iofs-pfs.so** to start one driver instance per MTP device, rather than using a single instance to service many devices. You can provide a separate `device` option with a unique set of values for each MTP device you want to support. For example, you can run these commands to support two MTP devices:

```
# io-fs-media -dpfs,device=1:3:3
# io-fs-media -dpfs,device=2:4:6
```

Bus, device, and interface numbers are hexadecimal values.

`escape`

Specify characters to be escaped in filenames. The escape behavior is as follows:

- If you don't specify this option, **iofs-pfs.so** by default escapes `/` to `%2F` and `%` to `%25`.
- If you specify an empty string by using `escape=` or `escape=""`, no characters are escaped (not even `%`).
- For all other `escape` settings, **iofs-pfs.so** escapes each character in the string. For instance, if you use `escape=" &\"` on the command line, the space character is escaped to `%20`, the ampersand (`&`) to `%26`, and the backslash (`\`) to `%5C`. Note that the order of the characters in the string doesn't matter; you could use `escape="\ &"` and the results would be the same.

`exitonconnerr`

Force **iofs-pfs.so** to exit after detecting an invalid MTP interface.

`mount=`*path*

Specify the root of the filesystem for MTP devices. If *path* ends in `$#`, then `$#` is replaced with a unit number. For instance, in a command line containing `-dpfs,mount=/fs/pfs$#`, the mountpoint could be replaced with `/fs/pfs0`.

`norefs`

Don't make files with references appear as directories. This setting disables playlists (e.g., **.alb** and **.pla** files on some devices) but can significantly reduce synchronization time.

`slabel`

Define the string for labelling the storage when it's presented in a list at the device mountpoint. This option is applicable when the `storage` option is set to `auto` or `all`. The supported settings are:

Setting	Description	Example
`desc`	*(Default)* The storage description provided by the device (e.g., "Phone", "Internal", "SD Card").	**/fs/pfs0/Internal**
`id`	The MTP storage ID provided by the device.	**/fs/pfs0/10001000**
`label`	The storage label, usually assigned by the user to the device or filesystem.	**/fs/pfs0/RJA_Phone**

We recommend using the default (`desc`) setting because the storage names listed will match what users see when connecting the device to their machine.

storage

Set the policy for listing the available storages of a device at its mountpoint. The MTP driver supports these storage-listing settings:

auto

(Default)

If the device has only one storage, list its contents at the device mountpoint (e.g., **/fs/pfs0**); this is the same behavior as `first`. If it has multiple storages, list the contents of its storages in subdirectories of the mountpoint (e.g., **/fs/pfs0/Internal**); this is the same behavior as `all`.

first

List the contents of the "first" storage directly at the device mountpoint (i.e., not in a subdirectory). The first storage is the one first reported by the device upon connection. When a device has many storages, it's up to the device to pick which one to report first. In this case, additional storages won't be listed at the mountpoint.

all

List the contents of all storages at the mountpoint, no matter how many are attached. For example, suppose a device has an internal storage and an inserted SD card. With this setting, you would see two subdirectories at the mountpoint: **/fs/pfs0/Internal** and **/fs/pfs0/SD_Card** (or something similar depending on the `slabel` setting).

The default option, `auto`, is convenient but non-deterministic. For the same device, different contents could be displayed at its mountpoint depending on the status of its external storage. For instance, if the SD card has been removed from the slot, the contents of the internal storage will appear at the mountpoint. If the SD card is inserted in the slot, a storage list (i.e., subdirectories for all accessible storages) will appear at the mountpoint, where each storage can be browsed independently.

If you're synchronizing a device, this non-deterministic setting could cause a lot of database thrashing between connections if the user often toggles the state of their removable SD card.

For more deterministic behavior, you must set `storage` to one of the last two options (`first` or `all`). With `first`, the contents of the first storage are always listed directly at the device mountpoint. With `all`, the contents of all accessible storages are always listed in subdirectories of the mountpoint.

usbpath

Name the USB stack to use. The default stack is **/dev/io-usb/io-usb**.

Description:

The **iofs-pfs.so** module implements the driver for MTP devices. This module is configured by issuing **io-fs-media** commands with the `-dpfs` option followed by a comma-separated list defining one or more driver options. For example, you can tell the MTP driver to list all storages at the device mountpoint, using user-defined labels to identify each storage, by issuing this command:

```
io-fs-media -dpfs,storage=all,slabel=desc
```

Enabling MTP:

To read files on MTP devices, you must configure the **usblauncher** service to start the **iofs-pfs.so** driver when an MTP device is connected. The **usblauncher** configuration contains rules that specify how to process specific devices when they're attached through a USB port. In the main configuration file (**rules.lua**), you must uncomment the rule containing the “MTP” Microsoft descriptor, by removing the "--[[" and "--]]" lines around the rule:

```
-- generic MTP rule should proceed generic MSC (mass storage) rule so that it is tried first.
--[[
-- example commented out
device() {
        ms_desc("MTP") {
                driver"io-fs-media -dpfs,device=$(busno):$(devno):$(iface)";
        };
};
--]]
```

To read media metadata (e.g., album name, artist name, track durations) from files on MTP devices, you must configure the metadata provider library (**libmd**) to use the appropriate plugin when accessing MTP files. You can do this by uncommenting the following lines in the **libmd** configuration file (**/etc/mm/mm-md.conf**), by removing the comment delimiter (#):

```
#[plugin]
#dll=mm-mdp-mediafs.so
...
#mtp=mediafs
```

These configuration changes must be done before starting the following components:

- **usblauncher**
- **mm-sync**
- **mm-renderer**