System Services 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 01, 2015

Contents

About This Reference

The *System Services Reference* lists the main services available in the QNX SDK for Apps and Media (also referred to as the Apps and Media SDK). This reference describes each of these services and, where applicable, how to configure and run them.

To find out about:	See:
Launching apps	*Application Launcher (launcher)* (p. 9)
Routing audio streams and managing the behavior of concurrent audio streams	*Audio Management* (p. 11)
Authorizing access to services and resources	*Authorization Manager (authman)* (p. 13)
Accessing camera video	*Camera Video (rearview-camera)* (p. 21)
Determining geo-coordinates	*Geolocation* (p. 33)
Generating images (`mksysimage.py` and other utilities)	*Image Generation Utilities* (p. 35)
Managing the on-screen and USB keyboards	*Keyboard (keyboard-imf)* (p. 59)
Setting up networking	*Network Manager (net_pps)* (p. 63)
Shutting down	*Shutdown (coreServices2)* (p. 65)
Managing the launch order of processes at startup	*System Launch and Monitor (SLM)* (p. 69)

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
Application Launcher (`launcher`)

Take launch requests from the HMI and check app permissions with the Authorization Manager

Synopsis:

```
launcher [-b]
         [-k]
         [-l]
         [-m app_mem_limit]
         [-p path prefix]
         [-s service prefix]
         [-t num_apps]
         [-U uid[:gid,sgid,...] ]
         [-v]
         project_name
```

Runs on:

QNX Neutrino

Options:

`-b`

Disable background launch (i.e., launch in the foreground).

`-k`

Disable background verify.

`-l`

Disable lock of memory pages for **launcher**.

`-m` *app_mem_limit*

RLIMIT_FREEMEM *rlim_cur* value (e.g., 970 is 97.0%).

`-p` *path prefix*

Location of target root (default is **/**).

`-s` *service prefix*

Location of target **pps** (default is **/pps/services/launcher**).

`-t` *num_apps*

Number of most-used apps to batch-verify before starting window manager.

-u ***uid*[:*gid*,*sgid*,...]**

Set explicit *uid* and optionally *gid* and any number of *sgid*s.

-v

Increase output verbosity. The -v option is cumulative; each additional v adds a level of verbosity.

Description:

The launcher service acts as a kind of go-between, taking requests from the HMI to launch an app while at the same time checking with the *Authorization Manager* (p. 13) (authman) to confirm that the app has the appropriate permissions to do what it wants.

The service communicates with the rest of the system using the **/pps/services/launcher/control** object. For more information about PPS objects, see the "*PPS Objects Reference*".

PPS objects:

To launch an app, the launcher service echoes a start command to the launcher control object. This command contains the application's ID string, display parameters, and a numeric identifier:

```
echo "msg::start\ndat::application_id,display_parameters\nid::ID_number"
> /pps/services/launcher/control
```

You can manually launch apps by issuing the same start command. In this case, we strongly recommend that you provide the WIDTH and HEIGHT parameters to define the dimensions of the app display area (in pixels). These parameters ensure that the app displays correctly.

After the app starts, the service stores the process ID (PID) in the response message, which it writes to the same object. For information about working with this object, see the "**/pps/services/launcher/control**" entry in the *PPS Objects Reference*.

Chapter 2
Audio Management

The QNX SDK for Apps and Media uses the audio manager to route audio streams to output devices and to manage the behavior of concurrent audio streams.

Overview:

The QNX SDK for Apps and Media supports diverse audio stream types (sound effects, ringtones, media from a media player, etc.). Each audio stream type must be routed to the most appropriate output device (speakers, headphones, etc.). When two or more audio streams request access to an output device, applications must know which audio stream takes priority, and what should be done with the other audio streams so that:

- the audio stream with the highest priority takes precedence and has access to the preferred output device(s)
- other audio streams are attenuated or muted, as required by the system configuration
- media playback is stopped or paused when an audio stream is “ducked” in favor of a higher priority audio stream, and restarted when appropriate

Applications implemented using QNX SDK for Apps and Media should use the audio manager to route audio streams to their preferred output devices, and based on the configuration, attenuate or mute audio streams when higher-priorty audio stream types open. The audio manager doesn't pause or stop audio playback. For example, the audio manager may mute media playback when the telephone rings, but the media will continue playing. Applications should use the Now Playing service to manage pausing and stopping playback.

Audio manager:

The audio manager and its library of functions and data structures provide:

- automatic routing, and manual routing of the PCM *preferred* path
- audio stream type identification
- audio concurrency policy (ducking) management
- audio device monitoring and mounting (e.g., headset, A2DP, HDMI)

For more information about using the audio manager, see *Audio Manager Library Reference*.

PPS objects:

The audio manager uses the PPS objects listed below. For information about these objects, see the relevant pages in the *PPS Objects Reference*.

- **/pps/services/audio/audio_router_control**
- **/pps/services/audio/audio_router_status**
- **/pps/services/audio/control**
- **/pps/services/audio/devices/**
- **/pps/services/audio/status**
- **/pps/services/audio/types/**
- **/pps/services/audio/voice_status**

Chapter 3
Authorization Manager (`authman`)

Indicate whether a process has access to requested system services or resources

Synopsis:

`authman [-b] [-c] [-U` *uid*`[:`*gid*`]] [-v]`

Runs on:

QNX Neutrino

Options:

`-b`

Disable background launch (i.e., launch in the foreground).

`-c`

Specify the configuration file to use for running `authman`. The default is **/etc/authman/authman.cfg**.

`-U` *uid*`[:`*gid*`]`

Run with the specified credentials when possible, to avoid running as root all the time. When started, `authman` runs as root to perform privileged operations. If you use the `-U` option, the service drops its credentials to the user ID (*uid*) and the group ID (*gid*), if it's provided. The argument format must match the format in the **/etc/passwd** and **/etc/group** files.

`-v`

Increase output verbosity. The `-v` option is cumulative; each additional `v` increases the level of verbosity, up to three levels (`-vvv`).

Description:

The Authorization Manager (`authman`) is a *resource manager* that handles requests from other processes to access system services they may need, such as the PPS filesystem, system paths, permissions, or OS system calls. The `authman` service uses a set of capabilities to protect system services from unauthorized use. Capabilities specify the apps that can use a particular service.

Although `authman` is responsible for determining whether an app can use the services it wants to use, the app doesn't send requests directly to `authman`. Instead, the *Application Launcher* (p. 9) (`launcher`) does this on the app's behalf. When asked to launch an app, the `launcher` process asks `authman` to confirm that the app has permission to use the requested capabilities.

Launcher is responsible for handling situations when the app isn't allowed access to the requested resources. If Launcher allows the app to run without certain capabilities, the app is responsible for handling how to run without having access to all the resources that it requires.

The authorization process is as follows:

1. When an app is packaged, its **MANIFEST.MF** file contains any capabilities that were listed in the <action> element in the **bar-descriptor.xml** file.
2. When it receives a request to launch an app (e.g., from the HMI), the launcher process reads the app's **MANIFEST.MF** file for the requested capabilities (e.g., Entry-Point-System-Actions: access_shared).
3. The launcher process asks authman to confirm that the app is entitled to do what it wants to do.
4. The authman process checks the **sys.res** file to see if the app has an allow permission for each requested capability.
5. If authman returns true for the capability request, launcher starts the app and the app can access all the resources it needs. If authman returns false, launcher can still start the app, but the app must run with fewer resources.

Files used for authorization

The following files are used when the system attempts to launch an app:

File	Description
/apps/*name*/native/bar-descriptor.xml	A configuration file that accompanies the app's *BlackBerry ARchive* (BAR) file, which contains all the app's code and resources. The **bar-descriptor.xml** file lists an app's assets, window attributes, capabilities (given in the <action> element), and more. This file is primarily used by the tools, which use its information to generate the manifest file, **MANIFEST.MF**.
/apps/*name*/META-INF/MANIFEST.MF	Generated during packaging, the **MANIFEST.MF** file contains various identifiers for the app, as well as the capabilities it wants to access (e.g., access_internet).
/etc/authman/sys.acl	Lists all the capabilities and their associated ACL (access control list) filesystem permissions. The authman process reads this file to determine whether an app has the permissions it needs.
/etc/authman/sys.res	Lists the available system capabilities and the apps that are entitled to use them. The authman process checks this file before authorizing an app to be launched.

Format for the sys.acl file

This file lists all the available capabilities, along with the filesystem permissions for particular paths that apps may use. These paths are typically PPS directories, but you can specify any paths.

To ensure your authorization rule changes are permanent, modify the **sys.acl** file on your host or development system. This file is found in ***$QNX_DEPLOYMENT_WORKSPACE*/target/etc/authman/**. Any changes made to the **sys.acl** copy on the target are lost when you rebuild and redeploy the image. You should refer to the *Getting Started* guide for the value of the *QNX_DEPLOYMENT_WORKSPACE* variable.

If a capability isn't listed in this file but an app requests that capability, the app is given access to it, based on the default `authman` behavior. Keep this default behaviour in mind to properly secure your system.

To define a capability, you must list its name on a separate line and list any actions that apply to it on subsequent lines, which must be indented. The general format is as follows:

```
capability_name
    ACTION  flags  [arguments]*
    ...
```

The *capability_name* field is a string that stores the capability name. The **authman** service supports these actions:

ACL (Access Control List)

Assigns permissions to paths accessible with that capability

MAC (Macro)

Invokes a macro, which is a group of actions

PAB (Procmgr Ability)

Defines *procmgr abilities*, which control the operations that applications are permitted to do (if they're granted the enclosing capability)

An action can be followed by certain flags, depending on the action. The **authman** service supports these flags:

opt

Indicates the action is optional, meaning if it fails, the other actions for the capability can still be applied. Without this flag, failure of the action causes **authman** to roll back the entire capability.

override_gid

Apply the action to this group ID (GID) instead of the app's GID. Overriding the GID on an action is useful when a service needs to create internal processes with unique GIDs.

Access control lists

Each access control list (ACL) action applies a permissions mask to a particular path:

```
capability_name
    ACL [opt]  group_mode[:other_mode] [path]
```

where:

opt

Specifies that this ACL is optional to the capability. If this option isn't used and the path subsequently specified isn't available, the permission is rolled back. The `opt` string is useful when a path might not be present. For example, when a PPS object gets created later.

group_mode[:other_mode]

Specifies group and optionally, other permissions for the path that follows. Each mode can contain any combination of the read (r), write (w), and execute (x) permissions.

By default, **authman** clears all the other bits to control path access strictly by group, but you can retain those bits by defining them in this flag. This option is useful for retaining execute permissions on a directory when a write ACL is applied. It's also useful for making a file readable for everyone, but writeable only by the ACL.

path

Provides the filepath or directory to access. It's common to point to a PPS object path, but it can be any filepath or location.

For example:

```
read_geolocation
    ACL opt rw /pps/services/geolocation/control
    ACL rw /pps/services/geolocation/status
```

The entry above indicates that any app that wants to use the `read_geolocation` capability has read and write permissions on the **/pps/services/geolocation/control** and **/pps/services/geolocation/status** objects. If **/pps/services/geolocation/control** doesn't exist, access to **/pps/services/geolocation/status** still applies, provided it's available.

Macros

You can define macros to group actions together and to reuse them in multiple capabilities and for many GIDs. A macro must be defined before its first usage in a capability (i.e., earlier in the configuration file), as follows:

```
macro macro_access_deviceproperties_private
    ACL x /pps/services/private
    ACL r /pps/services/private/deviceproperties
```

The entry above defines the `macro_access_deviceproperties_private` macro, which groups two ACL actions. These actions allow apps to navigate to the **/pps/services/private** directory (by assigning it the `x` permission) and to read the **/pps/services/private/deviceproperties** file (by assigning it the `r` permission).

You can use the MAC action to invoke actions from another capability as follows:

```
read_device_identifying_information
    MAC macro_access_deviceproperties_private
    MAC macro_access_device_properties
```

The `read_device_identifying_information` capability performs the actions in the `macro_access_deviceproperties_private` and `macro_access_device_properties` macros. Although not shown here, the `opt` flag can be specified for a MAC action; this way, if any action in the referenced macro fails, the other capability actions can still be applied. You can also use the *override_gid* flag to apply a GID to the multiple paths defined by the ACLs within a macro.

When using macros, you must be careful not to create *duelling capabilities*, which occur when two capabilities apply an ACL to the same path. This causes problems because if one ACL grants access to a path while the other ACL denies access, the capability that was processed last is the winner.

Procmgr abilities

Procmgr abilities are process-manager settings that control which operations are permitted for a particular process. When given these abilities, a process obtains functionality normally restricted to root.

Using PAB actions, you can define procmgr abilities for a capability as follows:

```
capability_name
        PAB  ability_name  inheritance_flag
          [sr1_low-sr1_high  [...  sr5_low-sr5_high]]
```

The *ability_name* string following the `PAB` keyword indicates the ability being granted. The abilities supported by the Apps and Media SDK are listed in the *procmgr_ability()* description found in the *QNX Neutrino C Library Reference*. Note that this list provides the numeric constants that can be passed to the library function; in the **sys.acl** file, however, you must refer to abilities by using strings. The string name for an ability is the name of its numeric constant without the `PROCMGR_AID_` prefix. Also, **authman** reads the ability strings in a case-insensitive manner. This means you can use `child_newapp` in **sys.acl** to refer to the ability represented by `PROCMGR_AID_CHILD_NEWAPP`.

The *inheritance_flag* must be either `inherit_yes` (to ensure this new ability setting is inherited by child processes) or `inherit_no` (to ensure the setting isn't inherited, which is the default behavior).

Depending on the ability, you can define one or more value subranges (in the *srN_low* and *srN_high* fields). The units and meaning of these values are specific to the ability. For example, the `PROCMGR_AID_SPAWN_SETGID` and `PROCMGR_AID_SPAWN_SETUID` abilities accept lower and upper bounds on the GIDs or UIDs that a process can assign to a child process. The `PROCMGR_AID_MEM_LOCK` ability locks a range of the process address space into physical memory; in this case, the range arguments specify the start and end addresses.

The following capability rule defines multiple abilities that have different inheritance and subrange values:

```
is_launch_delegate
    PAB setuid inherit_yes 1-2147483647
    PAB setgid inherit_yes 1-2147483647
    PAB child_newapp inherit_no
    PAB pathspace inherit_no
```

The PAB statements shown above assign abilities to applications granted the `is_launch_delegate` capability. These applications can then assign any UID or GID to their child processes, as indicated by the maximum numeric range given for the `setuid` and `setgid` abilities. They can also create a new application ID for a child process (as indicated by `child_newapp`), and add items to the kernel pathname prefix space (as indicated by `pathspace`).

Format for the sys.res file

This file is used to restrict authorization—only the particular apps listed under each available capability can use that capability.

To ensure your authorization rule changes are permanent, modify the **sys.res** file on your host or development system, for the reason explained in the **sys.acl** file description.

If a capability is defined in the **sys.acl** file but not listed in **sys.res**, an app requesting that capability is automatically granted access to it (effectively `allow *`). It's important to remember this default behavior to properly secure your system.

The format to define access to a capability is as follows:

```
capability_name
    [allow|deny] [application-name|application-name*|*]
capability
```

Here's an example of how to allow any apps to use the `access_internet` capability:

```
access_internet
    allow *
```

In the example above, *any* is indicated by the wildcard (`*`).

Conversely, if you want specific apps to access a capability, you can define a *whitelist*. A whitelist gives access or privileges to those that are listed, but denies access to all other apps or processes. You can specify apps and processes using a partial name and wildcard (`*`) or using a full name. The names of apps can be determined when you install them on the target. For example, the following specification grants the `access_special_info` capability to apps with names beginning with *mybrowser* (e.g., `mybrowser_one`) and to an app with the full name *mydemo.testDev_emodemod448125f* :

```
access_special_info
    deny *
    allow mybrowser*
    allow mydemo.testDev_emodemod448125f
```

Predefined capabilities

After a capability is granted, it allows an app to use a service that would be otherwise restricted. This consideration is important when you are designing capabilities for your system. You can define any number of capabilities. In addition, your host and target image may come with predefined capabilities that you can use and build into your image. Here are the capabilities provided for the QNX SDK for Apps and Media:

Capability	Description
`access_demo`	Do not use in production systems. This capability is used for the Cordova PPS Demo sample, which shows how to restrict application access to a PPS object and how to define a whitelist. For information

Capability	Description
	about this permission, see "Adding a permission to access a PPS object" in the *HTML5 Developer's Guide*.
access_internet	Use the internet connection from a Wi-Fi, wired, or other type of connection to access locations that are not local on the target.
access_shared	Read and write files shared between applications that run on the target.
audio_manager_access	Access restricted Audio Manager capabilities that include, but aren't limited to, adjusting volume and voice configuration features. For more information, see the functions in the Audio Manager API.
configure_system	Configure system-level functionality, such as network settings, restarting, or shutting down of the target.
read_geolocation or access_location_services	Read the target's current location. This location is determined based on the assigned IP address.

The capabilities listed below may be found in the **sys.acl** and **sys.res** files installed on your host. These capabilities are required for authman to function correctly but are for internal use only. Don't remove any of the permissions below, as it may cause undefined behavior:

- access_bbid
- access_pimdomain_enterprisecalllogs
- access_pimdomain_enterprisecontacts
- access_pimdomain_personalcontacts
- access_remote_framewriter
- access_social_lookup
- access_sys
- allow_app_purchase
- apkruntime
- is_apkruntime_launch_delegate
- is_launch_delegate
- list_pimdomain_enterpriseaccounts
- list_pimdomain_personalaccounts
- manage_cert
- play_audio
- post_notification
- read_device_identifying_information
- record_audio
- request_pim_session
- run_air_native
- set_audio_volume
- use_camera
- use_installer

Configuration file

The **authman** configuration file provides additional controls for enabling access to system services, based on process permissions and capability names. For instance, you can define *client whitelists* to restrict the commands that client applications can issue to **authman**. These commands provide dynamic updates to your security configuration, such as setting and deleting capabilities for applications.

Each whitelist names the commands that clients with a certain set of credentials (i.e., process permissions) can use. Entries in this configuration section have the following form:

```
username= <username>, ["default"|override_uid], ["default"|override_gid],
 <cmd_1>, <cmd_2>, ..., <cmd_N>
```

When `"default"` is given for the UID or GID, the permission setting in **/etc/passwd** is used. You can override the permissions by defining the *override_uid* and/or *override_gid* values. Consider this excerpt:

```
[client_whitelist]
#         USERNAME        UID      GID      CMDS LIST
username= apps,           default, 0,       chkcap
```

This command ensures that clients with the username `apps`, running with their default UID and a GID of 0, can issue the `chkcap` command to the **authman** service.

You can also whitelist the paths that can be affected by rule files loaded at runtime. The owner of a *capability prefix*, which is used to represent a capability category, can send the `load` command to **authman** to dynamically load, from an **.acl** or a **.res** file, a set of capability-based pathname permissions or application access settings. In the configuration file, each whitelist statement contains a capability prefix followed by a path. The statement ensures that only those apps with a dynamically loaded capability that begins with the stated prefix can access the path that follows. Apps without capabilities based on this prefix can't access the path.

The default configuration file defines these prefix-based paths:

```
[capability_prefix_paths]
prefix=cds, /pps/services/networking/control
prefix=cds, /pps/services/networking/status
prefix=cds, /pps/services/networking/proxy
prefix=cds, /pps/services/wifi/status
```

These settings ensure that only apps that are granted capabilities with the `cds` prefix at runtime can access the stated PPS paths. The advantage of defining such whitelists is that they let you constrain the paths affected by certain capabilities without requiring someone with system expertise to review the rule files to ensure that they meet security requirements. Likewise, these files only define access settings and pathname permissions, for security reasons.

Chapter 4
Camera Video (`rearview-camera`)

Allow access to the video stream for the camera (connected using USB or directly to the target board)

Syntax:

```
rearview-camera -bsize=dimensions
                [-actvid-hsync=actvid]
                [-adaptive-dei-int-mode=mode]
                [-adaptive-dei-motion-mode=admode]
                [-brightness=level]
                [-clock-pol=polarity]
                [-color-test=colortesttype]
                [-contrast=level][-cpos=coordinates]
                [-csize=dimensions]
                [-data-bus-width=buswidth]
                [-device=deviceid]
                [-display=displayid][-ddr-clk=clock]
                [-dmode=deinterlacemode]
                [-edma-chan=channel]
                [-fid-pol=polarity][-fill=buffmem]
                [-format=bufferformat]
                [-frame-count=count][-frame-rate=rate]
                [-hsync-pol=polarity][-hue=color]
                [-interface-type=itype]
                [-nbufs=numbuffers] [-nlanes=lanes]
                [-norm=vstandard]
                [-parent-zorder=zorder]
                [-pipeline=pipelineid]
                [-pos=coordinates] [-prio=tpriority]
                [-quit-if-no-video] [-saturation=level]
                [-sensor-clk-mode=mode]
                [-sfsize=dimensions][-size=dimensions]
                [-spos=coordinates] [-ssize=dimensions]
                [-source=index] [-source-type=sourcetype]
                [-sync-type=stype]
                [-verbosity] [-video-info=vidinfo]
                [-vsync-pol=polarity]
```

Runs on:

QNX Neutrino

Options:

-brightness=*level*

An integer value in the range of -128–127 that specifies the brightness level to use for capturing video.

-bsize=*dimensions*

(Required) A pair of integers that specifies the dimensions of the buffers. The dimensions are delimited by an x. For example, 1024x800. The same buffer size is used to capture video and render display.

-clock-pol=*polarity*

An integer value that controls whether the clock polarity is inverted for capturing video. You can use the following values to specify polarity:

- 1 — Polarity is inverted.
- 0 — Polarity is not inverted.
- -1 — Don't set polarity. Use the default polarity set on the camera.

-color-test=*colortesttype*

An integer value that represents a color test. The colors are tested for the camera. The color test type can be one of the following values:

- 1 — Test the contrast.
- 2 — Test the brightness.
- 3 — Test the saturation.
- 4 — Test the hue.

-contrast=*level*

An integer value in the range of -128–127 that specifies the level of contrast to use for capturing video.

-cpos=*coordinates*

A pair of integers that specifies the position to crop the captured video. The *X* and *Y* coordinates are delimited using a comma (,). For example 10,10.

-csize=*dimensions*

A pair of integers that specifies the crop size to use for the captured video. The dimensions are delimited by an x. For example, 790x593.

-data-bus-width=*buswidth*

An integer that specifies the width of the data bus for a parallel interface to use for capturing video.

-device=*deviceid*

An integer that specifies the index of the device to use for capturing video.

`-display=` ***displayid***

A string that represents a numeric value or a display type. As a numeric value, it's used to represent the display ID. As a string value, it represents the display type using one of these values, which are case-sensitive:

`internal`

An internal connection type to the display.

`composite`

A composite connection type to the display.

`svideo`

An S-Video connection type to the display.

`YPbPr`

The YPbPr signal of the component connection.

`rgb`

The RGB signal of the component connection.

`rgbhv`

The RBGHV signal of the component connection.

`dvi`

A DVI connection type to the display.

`hdmi`

An HDMI connection type to the display.

`other`

A connection type to the display which is one other than internal, composite, S-Video, component, DVI, HDMI, or DisplayPort.

`-dmode=` ***deinterlacemode***

A string that specifies the de-interlace mode to use for the captured video. The mode can be specified using one of the following values, which are case-sensitive:

`adaptive`

Use motion adapative de-interlacing mode.

For Texas Instruments Jacinto 6 and Freescale i.MX6x SABRE Smart Device targets, setting this value causes the motion-adaptive de-interlacer (hardware) to be used.

`bob`

Use BOB de-interlacing mode.

bob2

Use alternate BOB de-interlace mode.

none

Don't de-interlace the video. This value is used if this option isn't set.

weave

Use WEAVE de-interlace mode.

weave2

Use alternate WEAVE de-interlace mode.

-fid-pol=*polarity*

An integer that controls whether the field ID polarity signal is inverted for captured video. You can use the following values to specify polarity:

- 1 — Polarity is inverted.
- 0 — Polarity is not inverted.
- -1 — Don't set polarity. Use the default polarity set on the camera.

-fill=*buffmem*

An integer that specifies what the buffer is initialized with.

-format=*bufferformat*

A string that specifies the buffer format. If this option isn't set, the default format is `yvyu`. The format can be specified using one of the following values, which are case-sensitive:

rgb888

RGB888 format.

uyvy

UYVY format.

yuy2

YUY2 format.

yvyu

(default) YVYU format.

-frame-count=*count*

An integer that specifies the total frame count to be captured by the camera.

-frame-rate=*rate*

A float that specifies the expected frame rate (in frames per second) for capturing video.

-hsync-pol=*polarity*

An integer that controls whether the horizontal sync polarity is inverted. You can use the following values to specify polarity:

- 1 — Polarity is inverted.
- 0 — Polarity is not inverted.
- -1 — Don't set polarity. Use the default polarity set on the camera.

-hue=*color*

An integer in the range of -128–127 that specifies the color used for capturing video.

-interface-type=*itype*

A string that specifies the type of interface used by the camera. The following values can be used, which are case-sensitive:

csi2

The interface is a MIPI CSI2 interface.

parallel

The interface is parallel.

-nbufs=*numbuffers*

An integer that specifies the number of buffers created by the application. You should allocate at least four buffers, but if you are using adaptive de-interlacing mode (specified using the `-dmode` option), allocate at least ten buffers.

If this option isn't set, ten buffers are allocated when adaptive de-interlacing is used; otherwise, four buffers are allocated.

-nlanes=*lanes*

An integer that specifies the number of CSI2 data lanes to be used for the camera.

-norm=*vstandard*

A string that specifies to use a National Television System Committee (NTSC), Phase Alternating Line (PAL), or Sequential Color with Memory (SECAM) video standard. For more information about:

- NTSC, see *https://en.wikipedia.org/wiki/NTSC*
- PAL, see *https://en.wikipedia.org/wiki/PAL*
- SECAM, see *https://en.wikipedia.org/wiki/SECAM*

The following video standard to capture video with can be specified using these values, which are case-sensitive:

NTSC_M_J

Standard used in United States and Japan.

NTSC_4_43

A pseudo-color system that transmits NTSC encoding (not a broadcast format).

PAL_M

PAL format that uses 525 lines and 59.94 fields per second; this video standard is used in Brazil.

PAL_B_G_H_I_D

PAL format using 625 lines and 50 fields per second with various signal characteristics and color encodings.

PAL_COMBINATION_N

PAL format with narrow bandwidth that's used in Argentina, Paraguay, and Uruguay.

PAL_60

Multi-system PAL support that uses 525 lines and 60 fields per second (not a broadcast format).

SECAM

Video standard used mainly in France.

-parent-zorder=*zorder*

An integer that specifies the z-order of the parent window. The video rendering window is an embedded window (child) of the parent window.

-pipeline=*pipelineid*

An integer that specifies the WFD pipeline ID. When an ID is specified, it indicates to the Screen Graphics Subsystem to use a non-composited layer on the display. The number of pipelines vary for each hardware platform and its behavior depends on a number of factors used by Screen. For information about the use of pipelines for composition, see "Understanding composition" in the *Screen Graphics Subsystem Developer's Guide*.

-pos=*coordinates*

A pair of integers that specifies the position of the video window on the display. The *X* and *Y* coordinates are delimited using a comma (`,`). For example, `10,10`.

-prio=*tpriority*

An integer that specifies the priority of the capture thread relative to the original thread priority. The default relative priority is +20.

-quit-if-no-video

Stop `rearview-camera` when no video input is detected. Depending on the decoder that's used, there may be a delay in detecting whether video input has stopped.

-reenable-delay=*delaytime*

An integer that specifies the delay time (in milliseconds) to enable video capture after it has stopped capturing the number of frames specified by the -frame-count= option. If this option is not specified when the -frame-count= option is specified, a prompt appears on command line of the target to click the Enter key to continue video capture.

-saturation=*level*

An integer in the range of -128–127 that specifies the intensity level of color to use for capturing video.

-sfsize=*dimensions*

A pair of integers that specifies the dimensions of the source frame used for capturing video. The dimensions are delimited by an x. For example, 720x400.

-size=*dimensions*

A pair of integers that specifies the dimensions of the rectangle to show the video on the display. The dimensons are delimited using an x. For example, 720x400.

-ssize=*dimensions*

A pair of integers that specifiy the dimensions of the source viewport used for the captured video. The dimensions are delimited by an x. For example, 1024x800.

-source=*index*

An integer that specifies the index of the device's video capture unit.

-spos=*coordinates*

A pair of integers that specifies the position of the source viewport. The *X* and *Y* coordinates are delimited using a comma (,). For example, 10,10.

-ssize=*dimensions*

A pair of integers that specifies the dimensions of the source viewport used for the captured video. The dimensons are delimited using an x. For example, 1024x800.

-verbosity

An integer that specifies the verbosity level for debugging.

-video-info=*vidinfo*

An integer that specifies the number of frames to wait before checking for the video information, such as NTSC_M_J or PAL_M.

-vsync-pol=*polarity*

An integer that controls whether the vertical sync polarity is inverted. You can use the following values to specify polarity:

- 1 — Polarity is inverted.
- 0 — Polarity is not inverted.

- -1 — Don't set polarity. Use the default polarity set on the camera.

The following options are available only for specific platforms:

Platform	Options
Texas Instruments Jacinto 6	**`-actvid-hsync=`*actvid*** An integer that specifies to use ACTVID style line capture. A value of zero or one can be used to specify the ACTVID style. The default is negative one (-1) when this option isn't set. **`-adaptive-dei-int-mode=`*mode*** A string that specifies the interpolation mode for the adaptive de-interlacer. The mode can be specified using one of these values, which are case-sensitive: **`mode0`** The interpolated field is created by line averaging from the YUV data. That is, the interpolated line is created by averaging its top and bottom lines. **`mode1`** The interpolated field is created by averaging pixels from fields before and after the current field. For example, if the current field is a top field, the interpolated bottom field is created by averaging pixels from bottom field pictures before and after the current field. **`mode2`** This mode is an edge assisted interlace mode with edges detected from the Luma information in the frame window. Luma for missing lines are interpolated using original Luma along the detected edge. MV from the MDT module is used to select coefficients from a LUT on how 2D interpolation from the current field and 3D interpolation from two fields adjacent to the current fields are blended. **`mode3`** The edge detection method used in this mode is similar to interpolation mode 2. The only difference is that the edge-directed interpolation is performed on both Luma and Chroma. Chroma is interpolated similarly according to the edge vectors obtained based on Luma information. This value is default if this option isn't specified. **`-edma-chan=`*channel*** An integer that specifies the enhanced direct memory access (EDMA) channel to use WEAVE2 de-interlacing. You can set a board-specific channel or use a default EDMA channel. Use a range from 0–63 to specify a board-specific channel; otherwise, use negative one (-1) to specify the default EDMA channel. When there's no EDMA channel specified, the default channel is used. The default EDMA channel number is bound to the specific device and source as follows: • channel 6 for device 0 and source 0 • channel 7 for device 1 and source 0

Platform	Options
	• channel 62 for device 0 and source 1 • channel 63 for device 1 and source 1 **-sync-type=*stype*** An integer that specifies the synchronization signal type. You can set from 0–15. Alternatively, set the value to negative one (-1) to use the default channel.
Texas Instruments OMAP5	**-ddr-clk=*clock*** An integer value that specifies the dual-data rate clock (MHz) of the external CSI2 transmitter.
Freescale SABRE i.MX6x	**-adaptive-dei-motion-mode=*admode*** A string that specifies the motion mode for the adaptive de-interlacer. The mode can be one of the following values, which are case-sensitive: **low** Do minimal conversion of interlaced fields to one progressive frame. **med** (Default) Use default settings for converting interlaced fields to a progressive frame. **high** Do as much processing as required to produce higher image quality. It may introduce some delays. **-sensor-clk-mode=*mode*** An integer that specifies the sensor clock mode using one of the following values: • 0 — Gated clock. • 1 — Non-gated clock. • 2 — Progressive video interface CCIR 656. • 3 — Interlaced video interface CCIR 656. • 4 — Progressive video interface CCIR 1120 using double data rate. • 5 — Progressive video using standard data rate. • 6 — Interlaced video interface CCIR 1120 using double data rate. • 7 — Interlaced video interface CCIR 1120 using standard data rate.

The following options are available only for specific video decoders:

Video decoder	Options
Analog Devices ADV7x	**-source-type=*sourcetype*** An integer that specifies the input source type. This option should be set when you are setting the `-source` option. You can use the following source types:

Video decoder	Options
	• 0 — Single-ended composite (CVBS). • 1 — Sequential Color with Memory (SECAM). • 2 — Component interface (pCBCr). • 3 — Differential composite (CVBS).

Description:

The `rearview-camera` utility starts the rearview-camera service. This service creates a parent window and an embedded window. The embedded window joins the parent window, which is visible, so that the camera stream can be seen on the display. After the HMI is running, it creates a window that the embedded window can join and drops the original parent. The HMI creates an entry in a PPS object so the rearview-camera service knows what window group to join.

To start the service manually, run the `rearview-camera` command located at **/base/usr/bin/rearview-camera**.

In the QNX Apps and Media image, this command is run from the **camera-start.sh** startup script.

The service creates raw images that are encoded with the following syntax:

format-*width*x*height*-*stride*-#*seqno*-**rearviewcapture.raw**

Before you can use the rearview-camera service, the touchscreen connected to your target must be calibrated using the `calib-touch` utility. After the touchscreen is successfully calibrated, you can use `calib-touch` to save a configuration file to **/etc/system/config/calib**.

The rearview-camera service is built using Screen Graphics Subsystem to display video and is captured using the Video Capture API Reference. For information about Screen Graphics Subsystem, see Screen Graphics Subsystem Developer's Guide and Video Capture Library Reference.

PPS objects:

The service uses the following PPS objects:

- **/pps/system/navigator/windowgroup**
- **/pps/system/navigator/command**

The rearview-camera service joins the window group that the HMI identifies in the **windowgroup** object using the `rearview_camera` attribute:

```
[n]rearview_camera::{94121bb8-4122-4dfb-8321-181eaf4c2553}
```

The rearview-camera service also listens to the **command** object and looks for either a `pause` or `resume` action, which represents its new state:

```
rearview_camera:json:{"action": "pause"}
rearview_camera:json:{"action": "resume"}
```

Based on whether it enters the `pause` or `resume` state, the service stops or starts rendering video.

Configuration file:

The camera is called from a startup script located at **/scripts/camera-start.sh**. The script checks a configuration file located at **/var/etc/services-enabled**. An attribute called USBCAM can be set to `true` if a USBCAM is used on the target board; otherwise, it's set to `false`. When a USBCAM is available, the USB version of the *libcapture* library should be linked to **/base/usr/lib/libcapture.so.1**; otherwise, the *libcapture* library for the target should be linked to **/base/usr/lib/libcapture.so.1**.

For example, to link the USB version of the library, use the following lines in your startup script on the target:

```
ln -Psf /base/usr/lib/libcapture-usb-uvc.so
        /base/usr/lib/libcapture.so.1
```

Otherwise, link the target-specific *libcapture* library:

```
ln -Psf /base/usr/lib/libcapture-board-imx6x-sabreSMART.so
        /base/usr/lib/libcapture.so.1
```

Examples:

To start USB camera for an existing HMI on an OMAP5432 EVM target:

```
rearview-camera -zorder=-1 -parent-zorder=-1 -pipeline=1 -format=yuy2
                -sfsize=640x480 -bsize=640x480 -ddr-clk=384 -source=0
                -nlanes=1
```

To start video capture with UYVY format, without any de-interlacing bound to pipeline four:

```
rearview-camera -format=uyvy -pipeline=4 -dmode=none -bsize=720x240
```

To start video capture with YVYU format and adaptive de-interlace mode using interpolation mode 3, bound to pipeline four, and video cropping size set to 720x400:

```
rearview-camera -bsize=720x480 -pipeline=4 -dmode=adaptive
                -adaptive-dei-int-mode=mode3 -csize=720x400
```

Some formats aren't supported by the service's encoding formats. For instance, FFmpeg is a format that isn't supported and requires that you switch certain bits. You are responsible for handling the raw format conversion. Often, this requires that you develop your own tools to convert the raw image to another image format. For example, you might need to run the raw format through a tool as shown here:

```
cat uyvy-720x480.raw | ffmpeg -vcodec rawvideo -f rawvideo
                       -pix_fmt uyvy422 -s 720x480 -i pipe:0
                       -f image2 -vcodec png uyvy-720x480.png
```

In this case, after you output the raw format to create an image, additional processing is required. You may need to write a tool to handle this processing.

Chapter 5
Geolocation

The Geolocation service provides the current location of the client based on its IP address.

Upon receipt of a `location` request message from the client, the Geolocation service queries *http://www.hostip.info* to get the current location, based on the client's IP address. The correctness of the result depends on the contents of the database that **hostip.info** provides. If the client's IP isn't in the database, an incorrect location might be returned.

Client queries about location information can be made using the following PPS command:

```
(exec 3<>/pps/services/geolocation/control?wait &&
echo 'msg::location\nid::test\ndat:json:
{"period":5.0,"provider":"network","fix_type":"wifi"}' >&3 &&
cat <&3)
```

where:

- `period` specifies the interval between updates from the server. If the period is 0, only one update is sent.
- `provider` specifies the type of connection. In this case, a network is used to determine the location.
- `fix_type` specifies the type of connection. A value of `wifi` indicates that it's a valid wireless or cable-connected network; it doesn't indicate that the connection is Wi-Fi.

The Geolocation service responds to the client in the following format in the `control` object:

```
@control
res::location
id::test
dat:json:{"accuracy":60,"latitude":45.3333,"longitude":-75.9}
```

The browser also uses the Geolocation service to query the location as shown above. The Geolocation service is multithreaded and can handle requests from multiple clients at the same time.

PPS objects:

For more information about the PPS objects that the Geolocation service uses, see these entries in the *PPS Objects Reference*:

- **/pps/services/geolocation/control**
- **/pps/services/geolocation/status**

Chapter 6
Image Generation Utilities

The QNX SDK for Apps and Media includes tools to assist you with building images for your target.

The tools included with the QNX SDK for Apps and Media are in addition to those available with the QNX Neutrino RTOS SDP. For information about the SDP utilities, see the *Utilities Reference*.

For information about building images, see:

- "How to create a target image" in *Getting Started*
- *Building Embedded Systems* in the QNX SDP documentation
- The *BSP User Guide* for your board at *Foundry27*

diskimage

Create an image for a partitioned medium, such as a hard drive, SD card, or MMC

Syntax:

```
diskimage -c configfile -o imgfile
          [-b bootstrapfile]
          [-h]
          [-m] [-M]
          [-p] [-s number] [-v]
```

Runs on:

Windows, Linux

Options:

-b *bootstrapfile*

Use the specified bootstrap file to write to the master boot record (MBR).

-c *configfile*

Use the specified configuration file.

-h

Display a help message showing `diskimage` usage information.

-m

Don't modify Power-Safe (**fs-qnx6.so**) filesystems. The default is to patch all bootable Power-Safe filesystem partitions.

-M

Modify all Power-Safe filesystems, including non-bootable partitions.

-o *imgfile*

Write to the specified image file.

-p

Don't pad the image to the full disk size.

-s *number*

Read and write up to the specified number of bytes at a time. The default size is 4096.

-v

Increase the verbosity.

Description:

The `diskimage` utility creates an image for a partitioned medium. The partitioned medium image can contain any number of filesystem images. For example, Power-Safe filesystems that were created using `mkqnx6fsimg` or `mkqnx6fs`. After an image is successfully created, it can be copied to a hard drive, SD, or MMC. Options are available to adjust Power-Safe filesystems in the image so that they are bootable. With this command, you can also specify the IPL file to write into the master boot record (MBR).

You must use a configuration file to specify the disk image content and layout. The configuration file consists of a description for the disk and any number of optional descriptions about the partitions. There are three types of partitions you can use: primary, extended, and logical. The following constraints apply when you use `diskimage`:

- The disk can have up to four primary partitions or up to three primary and one extended partition. That extended partition can be used as a container for any number of logical partitions.
- Any logical partitions on the disk require the existence of an extended partition. When logical partitions are used, you're not required to explicitly specify the extended partition. Regardless of whether you do this, there can be only three primary partitions if you define logical partitions. If no extended partition is defined but logical partitions are defined, an extended partition is automatically created. You can't specify more than one extended partition in the configuration file.
- The extended partition must be explicitly defined when a specific partition index must be assigned to it or the required space exceeds the defined logical partitions. For information about calculating the size of the extended partition, see "*Using extended and logical partitions* (p. 40)".
- The combined size of all logical partitions (including any overhead) should be equal to and must not exceed the size of the extended partition.
- Filesystem images are copied to the disk image in blocks of up to 4096 bytes. You can override this default setting using the `-s` option.

Overview of configuration file syntax

The configuration file uses a specific structure, syntax, and grammar. For an explanation of the configuration structure (or contents), see "*Configuration file structure* (p. 37)". For a full description of the grammar, see "*Summary of configuration file syntax* (p. 42)". The configuration file can have these special characters:

- Whitespaces used in double quotes (`""`) of a string literal are interpreted as part of that string. Whitespaces aren't allowed within numeric literals (including the suffix).
- Line breaks and spaces can occur anywhere, except in literals or tokens.
- A line feed (`LF`) character indicates the end of a line. Carriage return characters (`CR`) are silently ignored and aren't interpreted as the end of line.
- A hash (`#`) character indicates the beginning of a comment. Comments can begin anywhere and always extend to the end of the line.
- Numeric values can be specified as decimal, octal, or hexadecimal. Optionally, numeric values can use these suffix values to indicate a factor:
  - `k` or `K` to represent kilo (1024)
  - `m` or `M` to represent mega (1024^2)
  - `g` or `G` to represent giga (1024^3)

Configuration file structure

The configuration file structure has two parts:

- disk configuration (Required) A description of disk-wide parameters.
- partition definitions (Optional) Any number of partition descriptions. The partition descriptions specify the partitions to create, as well as the contents and characteristics (attributes) of each partition.

The disk configuration is required in each configuration file. The configuration file defines a number of disk attributes that mainly describe the disk's geometry. All disk attribute definitions must be enclosed in square brackets (`[]`). A single pair of brackets can contain one or more attribute definitions; multiple attribute definitions within a pair of brackets must be separated by a whitespace.

Disk attributes

The `diskimage` command supports the following disk attributes:

`cylinders=`*number*

(Required) Specifies the number of cylinders on the disk. The value must be in the range of 1–4294967295 (2^{32}-1).

`heads=`*number*

(Required) Specifies the number of heads on the disk. The value must be in the range of 1–255.

`sectors_per_track=`*number*

(Required) Specifies the number of sectors on each track. The value must be in the range of 1–63.

`sector_size=`*number*

(Optional) Specifies the number of bytes per sector. The value must be in the range of 1–4294967295 (2^{32}-1). If not specified, a default value of 512 is used.

`start_at_cylinder=`*number*

(Optional) Specifies the cylinder at which the first partition shall begin. The value must be in the range of 0–4294967295(2^{32}-1). The default is zero.

The first track is always reserved for the MBR. Because of this, a partition beginning at cylinder zero has `sectors_per_track` x (`heads`-1) sectors available in the first cylinder.

The partition definitions are optional and you can specify as many as you require. You can define primary, extended, and logical partitions using the following syntax where:

partn_idx

A unique value in the range of 1–4.

partn_file

The pathname of the file containing the filesystem image for the partition. The pathname must be enclosed in double quotes (""), and can't exceed the host's maximum pathname length. The file size must not exceed the partition size. For filesystems that are not Power-Safe filesystems, the image file can be smaller than the partition (although a warning is issued).

- Primary partition:

```
[partition=partn_idx boot=true | false
                     type=number
                     num_sectors=number]  partn_file
```

Example:

```
# Power-Safe filesystem, > 2GB, bootable
[partition=2 boot=true type=179 num_sectors=4273290] "../fsi/qnx6-1.fsi"
```

- Extended partition:

```
[extended=partn_idx num_sectors=number]
```

Example:

```
[extended=2 num_sectors=8256]
```

- Logical partitions:

```
[logical type=number
         num_sectors=number
         ebr_sectors=number]  partn_file
```

Example:

```
# Power-Safe filesystem (~500MB)
[logical type=178 num_sectors=1060290] "../fsi/qnx6-2.fsi"
```

Partition Attributes

The following partition attributes are supported by `diskimage`:

boot=*true* | *false*

(Optional) Specifies whether the partition should be marked as bootable. A boolean value is used, where `true` indicates that the partition is bootable and `false` indicates that it isn't. The default value is `false`.

type= ***number***

(Required) Specifies the partition type. The value must be in the range of one to 255 and should match the type of the filesystem in the partition image file. Frequently used types include 11/12 (DOS FAT32) and 177/178/179 (Power-Safe filesystem). For more information about partitions and a list of partition IDs, refer to *http://en.wikipedia.org/wiki/Partition_type*.

num_sectors= ***number***

(Optional) Specifies the number of sectors to allocate for the partition. The value must be in the range of 1–4294967295 (2^{32}-1). If this attribute isn't specified, it's set to the smallest number of sectors required for the partition's filesystem image.

ebr_sectors= ***number***

(Optional) Specifies the number of sectors to be reserved in front of the logical partition. At least one sector is required for the EBR (Extended Boot Record) associated with the partition. The value must be in the range of 1–4294967295 (2^{32}-1). If this attribute isn't specified, exactly one track is reserved. For more information, see the `sectors_per_track` disk attribute.

Using extended and logical partitions

When logical partitions are used, an extended partition is required as well. It can be defined explicitly or created automatically by `diskimage`. The extended partition acts as a primary partition and serves as a container for the logical partitions. Within the extended partition, the logical partitions are laid out in the order in which they appear in the config file.

The total combined size of all logical partitions (including any overhead) must be less than or equal to the size of the extended partition. To determine the space required for the extended partition, include the following items in your calculation:

- the size of each logical partition
- each logical partition's `ebr_sectors`

By default, the `ebr_sectors` option is set to the disk's number of sectors per track. The size of the `ebr_sectors` option includes the EBR and any optional padding. It's recommended that the default of one track is used.

Attention Even though the EBR requires one sector, the entire track is allocated for it.

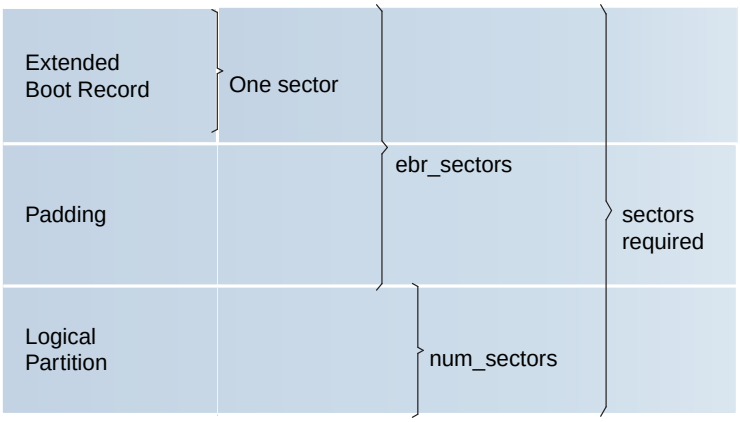

Figure 1: Extended boot record, padding, and logical partition

To illustrate the calculation, consider the following example:

In the following example, the specified partitions intentionally don't add up to fill the disk.

Target disk geometry

16 cylinders

32 heads

32 sectors per track

Partitions

One primary type 11 (DOS FAT32), bootable, 4096 sectors

One extended partition with:

- 1 logical type 179 (Power-Safe filesystem), 2048 sectors
- 1 logical type 178 (Power-Safe filesystem), 6144 sectors

To calculate the extended partition, use the following formula:

```
  <sectors_per_track> /* First logical EBR  */
+ <logical-type-179>  /* First logical data */
+ <sectors_per_track> /* Second logical EBR  */
+ <logical-type-178>  /* Second logical data */
= ext_sectors
```

Therefore, for the example, these are the values to use in the formula:

```
    32    /* sectors_per_track */
+ 2048    /* First logical data */
+   32    /* sectors_per_track  */
```

```
+ 6144    /* Second logical data */
= 8256    /* size of extended partition sectors */
```

To define the configuration specified in the previous example, the configuration file would be:

```
[cylinders=16 heads=32 sectors_per_track=32]
[partition=1 boot=true  type= 11 num_sectors=4096] "fat.img"
[extended=2                      num_sectors=8256]
[logical     boot=false type=179 num_sectors=2048] "qnx-179.img"
[logical     boot=false type=178 num_sectors=6144] "qnx-178.img"
```

However, if you wanted to presume that the three filesystem images are full-sized, you can use a more simplified syntax:

```
[cylinders=16 heads=32 sectors_per_track=32]
[partition=1 boot=true type=11] "fat.img"
[logical type=179] "qnx-179.img"
[logical type=178] "qnx-178.img"
```

Summary of configuration file syntax

The following excerpt explains the configuration file grammar:

```
# Configuration file
config_file : disk_cfg+ partn_def*

# Disk configuration
disk_cfg : '[' disk_attr+ ']'

# Disk attributes
disk_attr : 'cylinders' '=' uint
          | 'heads' '=' uint
          | 'sectors_per_track' '=' uint
          | 'sector_size' '=' uint
          | 'start_at_cylinder' '=' uint

# Partition definition
partn_def : primary_partn_def
          | extended_partn_def
          | logical_partn_def

# Primary partition definition
primary_partn_def :
      '[' 'partition' '=' partn_idx ppartn_attr* ']'
      partn_file

# Extended partition definition
extended_partn_def :
```

```
     '[' 'extended' '=' partn_idx epartn_attr* ']'

# Logical partition definition
logical_partn_def :
     '[' 'logical' lpartn_attr* ']'  partn_file

# Partition index
partn_idx : uint

# The primary partition attributes
ppartn_attr : boot_attr
            | type_attr
            | nsec_attr


# The extended partition attributes
epartn_attr : nsec_attr


# The logical partition attributes
lpartn_attr : boot_attr
            | type_attr
            | nsec_attr
            | esec_attr

# The boot attribute
boot_attr : 'boot' '=' bool

# The type attribute
type_attr : 'type' '=' uint

# The num_sectors attribute
nsec_attr : 'num_sectors' '=' uint

# The ebr_sectors attribute
esec_attr : 'ebr_sectors' '=' uint

# Partition file
partn_file : string

# The boolean definition
bool : 'true'
     | 'false'

# Unsigned integer
uint : '0' [0-7]+ sfx?
            | '0x' [0-9a-fA-F]+ sfx?
```

```
            | [1-9][0-9]* sfx?
# Factors that can be used to with integers.
sfx : [kKmMgG]
# string values must be enclosed within double quotes ("")
string : '"' [^"]* '"'
```

Examples:

Create an image

```
diskimage -c mydisk.cfg -o mydisk.img
```

Create an image and specify a primary boot loader (IPL)

```
diskimage -c mydisk.cfg
          -b C:\qnx660\target\qnx6\x86\boot\sys\ipl-diskpc1
          -o mydiskipl.img
```

Configuration file for three primary partitions

An example for a disk with three primary partitions, one DOS FAT32 and two Power-Safe filesystems. It's targeted at a disk geometry of 974/255/63 (cylinders/heads/sectors). The configuration file expects the filesystem images to reside in the directory `../fsi`.

```
[cylinders=974]
[heads=255]
[sectors_per_track=63]

# DOS FAT32
[partition=1
     boot=false
     type=11
     num_sectors=963837
] "../fsi/fat32.fsi"

# First Power-Safe filesystem, >2GB, bootable
[partition=2
     boot=true
     type=179
     num_sectors=4273290
] "../fsi/qnx6-1.fsi"

# Second Power-Safe filesystem
[partition=3
     boot=false
     type=178
     num_sectors=1060290
] "../fsi/qnx6-2.fsi"
```

Configuration file for two primary and two logical partitions

An example for a disk with two primary and two logical partitions. The primary partitions are DOS FAT32 and Power-Safe filesystems. Both logical partitions are Power-Safe filesystems. The extended partition uses slot 3 in the MBR. The resulting image is intended for a disk geometry of 4096/64/32 (cylinders/heads/sectors), which can represent an eMMC. The configuration file expects the filesystem images to reside in the directory `../fsi`.

```
[cylinders=4096 heads=64 sectors_per_track=32]

# DOS FAT32 (~480MB)
[partition=1
     type=11
     num_sectors=963837
] "../fsi/fat32.fsi"

# Primary Power-Safe filesystem, >2GB, bootable
[partition=2
     boot=true
     type=179
     num_sectors=4273290
] "../fsi/qnx6-1.fsi"

[extended=3]

# Power-Safe filesystem (~500MB)
[logical
     type=178
     num_sectors=1060290
] "../fsi/qnx6-2.fsi"

# Power-Safe filesystem (~500MB)
[logical
     type=177
     num_sectors=1060290
] "../fsi/qnx6-3.fsi"
```

Exit status:

0

Zero is returned when the command completes without errors.

1

One is returned when an error occurs while running the command. Possible errors include but aren't limited to:

- the configuration file wasn't specified
- the configuration file couldn't be read

- there were syntax errors in the configuration file

Caveats:

None.

gen-ifs.py

Perform setup activities and then run `mkifs` to include IFS files in an image. The mksysimage.py (wrapped by `mksysimage.sh` (Linux) or `mksysimage.bat` (Windows)) calls this command. You shouldn't call this command on its own.

Syntax:

```
gen-ifs.py  [-o] output [options]
```

Runs on:

Windows, Linux. Must be run using Python 2.7.5.

Options:

-c, --config

Read the specified IFS configuration file.

-d *defaultifs*, --default-ifs=*defaultifs*

Specify which IFS file is the default IFS (**qnx-ifs**).

--defaults

Include default directories in the search path.

-f *buildfile*, --input=*buildfile*

Include the specified input file.

-h, --help

Show the help that describes how to use this command.

-N--no-defaults

Don't include default directories in the search path.

-o *imagefile*, --output=*imagefile*

Write to the specified IFS file.

-P *productname boardname*, --product=*productname boardname*

Specify the product name and the board name. The entries that can be used for *productname* are the names of the subdirectories found in ***$QNX_DEPLOYMENT_WORKSPACE*/target/product/** and the possible entries for *boardname* are found in ***$QNX_DEPLOYMENT_WORKSPACE*/target/product/*productname*/boards/**. You should refer to the *Getting Started* guide for the value of the *QNX_DEPLOYMENT_WORKSPACE* variable.

--output-path

Write IFS(s) to the specified location.

-r *directory*, --root=*directory*

Include the specified directory as a root directory.

-v, --verbose

Increase verbosity.

Description:

The `gen-ifs.py` script defines *MKIFS_PATH* for the specified board type, locates and concatenates the specified buildfiles, and then runs `mkifs` to include the IFS files in an image. Don't use the `gen-ifs.py` script on its own. Instead, you should run *`mksysimage.py`* (p. 53), which calls the `gen-ifs.py` script.

An IFS is a bootable image filesystem that contains the `procnto` module, your boot script, and possibly other components such as drivers and shared objects. To create the IFS files to include in the final target image, `gen-ifs.py` calls the `mkifs` utility.

If you generate only a single IFS (**.ifs**) file, then you need to specify just the `-o` and `-f` options. However, if you want to generate multiple **.ifs** files, then you must specify at least the `-c`, `-d`, and `--output-path` options.

The following example uses the **omap5uevm** platform and creates an **.ifs** file. The information in the resulting file is used by the image-generation script *`mksysimage.py`* (p. 53).

Exit status:

0

The setup activities and inclusion of the IFS image files completed successfully.

>0

An error occurred.

Caveats:

None.

gen-osversion.py

*Generate the **/etc/os.version** file based on build environments. The mksysimage.py (wrapped by `mksysimage.sh` (Linux) or `mksysimage.bat` (Windows)) calls this command. You shouldn't call this command on its own.*

Syntax:

```
gen-osversion.py  [options] ... <platform>.<variant>
```

Runs on:

Windows, Linux. Must be run using Python 2.7.5.

Options:

`--branch`=BRANCH

Specify the branch to build.

`-d` GENERATIONDATE, `--date=GENERATIONDATE`

Specify the date entry. If this option isn't specified, the current time is used.

`-h, --help`

Show the help that describes how to use this command.

`-n` BUILDNUM, `--build-number`=BUILDNUM

Specify the build number. If this option isn't specified, the value in the *BUILD_NUMBER* environment variable is used.

`-q, --quiet`

Prevent output.

`-p, --additionalParameters`

Include additional parameters using the following notation <*parameter*>=<*value*>.

`-P` PRODUCT BOARDNAME, `--product`=PRODUCT BOARDNAME

Specify the product name and the board name. The entries that can be used for *productname* are the names of the subdirectories found in ***$QNX_DEPLOYMENT_WORKSPACE*/target/product/** and the possible entries for *boardname* are found in ***$QNX_DEPLOYMENT_WORKSPACE*/target/product/*productname*/boards/**. You should refer to the *Getting Started* guide for the value of the *QNX_DEPLOYMENT_WORKSPACE* variable.

`-r` REVISION, `--revision`=REVISION

Specify the revision of the image that's being built.

`-s` IMAGESCRIPT, `--image-script`=IMAGESCRIPT

Specify the script that was used to generate the image.

-u BUILDURL, --build-url=BUILDURL

Specify the build URL. If this option isn't specified, the value in the *BUILD_URL* environment variable is used.

--variant VARIANT

Specify the variant of the image that's being built.

-v, --verbose

Increase verbosity.

Description:

The `gen-osversion.py` utility to generates the **os.version** file based on the build environment. Don't use the `gen-osversion.py` script on its own. Instead, use the *`mksysimage.py`* (p. 53) script, which calls the `gen-osversion.py` script.

Exit status:

0

The OS version file was generated successfully.

>0

An error occurred.

Caveats:

None.

mkimage.py

Generate an image from existing ***.tar*** *files. The mksysimage.py (wrapped by* `mksysimage.sh` *(Linux) or* `mksysimage.bat` *(Windows)) calls this command. You shouldn't call this command on its own.*

Syntax:

```
mkimage.py  [-o] outputpath ... [options]
```

Runs on:

Windows, Linux. Must be run using Python 2.7.5.

Options:

`-b` *bootloader*, `--bootloader=` *bootloader*

Add the specified bootloader file to the master boot record (MBR).

`-c` *configfile*, `--config=` *configfile*

Use the specified configuration file. See "Configuration file for `mkimage.py`" in the *Getting Started* guide.

`-h`, `--help`

Show the help that describes how to use this command.

`-m`

Don't modify the bootable QNX6 filesystem.

`-o` *output*, `--output=` *output*

Write to the specified image file.

`-t` *tarfilepath*, `--tar_path=` *tarfilepath*

Specify the path to the **.tar** files.

`--tars` *tarfile1* [*tarfile2 ...*]

Specify a list of **.tar** files.

`-v`, `--verbose`

Specify the verbosity, up to a maximum of 4 levels.

Description:

The `mkimage.py` script creates an image from the **.tar** files, which are generated by `mktar.py`. Don't use the `mkimage.py` script on its own. Instead, you should run the *`mksysimage.py`* (p. 53) script, which calls the `mkimage.py` script.

The process used by `mkimage.py` to create an image is as follows:

1. Extract the contents of the input **.tar** files to a temporary location and from these contents, generate the buildfiles (**.build**), which list the files in the partitions.
2. Call `mkxfs` to use the buildfiles to generate the partition image (**.image**) files.
3. Call `diskimage` to combine the **.image** files into the disk image.

Exit status:

0

An image file was generated successfully.

>0

An error occurred.

Caveats:

None.

mksysimage.py

*Create an image of the platform and generate supporting files, such as **.ifs** and **.tar***

Syntax:

```
mksysimage.py  [-o output] [options]... [board_name.external]
```

Runs on:

Windows, Linux. Must be run using Python 2.7.5.

Options:

-c, --mksysimage-config-file

Specify the configuration file used for the `mksysimage.py` utility.

-f, --force

Force the overwriting of existing **.tar** files.

-g, --osversion-content

Specify any additional content for the **os.version** file.

-G, --no-gen, --no-generation

Run only `mktar.py` and the imaging components.

--gen-ifs-options

Specify the options for `gen-ifs.py` (see `gen-ifs.py --help`).

-h, --help

Show the help that describes how to use this command.

--image-config-path=*IMAGE_CONFIG_PATH*

Specify the path of the configuration files for `mkimage.py`.

-k *MKIMAGE_OPTIONS*, --mkimage-options=*MKIMAGE_OPTIONS*

Specify the options for `mkimage.py` (see `mkimage.py --help`).

-m *MKTAR_OPTIONS*, --mktar-options=*MKTAR_OPTIONS*

Specify the options for `mktar.py` (see `mktar.py --help`).

--no-gen-ifs

Don't generate **.ifs** files.

--no-mkimage

Don't run the `mkimage.py` part of the process.

--no-mktar

Don't run the `mktar.py` part of the process.

--no-gen-osversion

Don't generate an **os.version** file.

-o *OUTPUT_PATH*, --output-path=*OUTPUT_PATH*

Write **.image** and **.tar** files to the specified path. If the `-t` option is specified, the **.tar** files are written to that path instead.

-p, --keep-partition-images

Keep the partition images.

-P *PRODUCT*, --product=*PRODUCT*

Specify the name of the product. You can set the *QNX_PRODUCT* environment variable to specify the default product to use when this option isn't specified.

-q, --quiet

Prevent any output.

-Q *QT_PATH*, --qt=*QT_PATH*

Specify the location where Qt is installed on the host computer. You can set the *QNX_QT* environment variable to specify the default location to use when this option isn't specified.

-t, --tar-file-path

Read and write **.tar** files to and from the specified path.

-v, --verbose

Increase the verbosity.

-w *QNX_DEPLOYMENT_WORKSPACE*, --workspace=*QNX_DEPLOYMENT_WORKSPACE*

Specify the path to the *QNX_DEPLOYMENT_WORKSPACE*. This workspace is where you deployed the assets required to build the image.

You should refer to the *Getting Started* guide for the value of the *QNX_DEPLOYMENT_WORKSPACE* variable.

Description:

The `mksysimage.py` utility is a Python script that invokes other utilities to generate tar files and images for each platform. The script is located at ***$QNX_DEPLOYMENT_WORKSPACE*/infra/utils/scripts**. You should refer to the *Getting Started* guide for the value of *QNX_DEPLOYMENT_WORKSPACE*.

By default, `mksysimage.py` reads a configuration file from: ***$QNX_DEPLOYMENT_WORKSPACE*/infra/product/AnM/boards/*platform*.ext/mksysimage/*platform*-mksysimage.cfg**

This configuration file defines the tar files and images created during the image-generation process. The image variants for each platform are defined within the configuration file. By default, for each image variant, `mksysimage.py` generates two tar files and one image. The tar file ***platform*-os.tar**

contains two QNX filesystems that include all files except MLO and IFS files. The tar file ***platform*-dos-*image_variant*** contains a FAT16 filesystem that includes all bootup files, such as MLO and IFS files. The final generated image includes these two tar files.

You can change the default configuration file associated with `mksysimage.py`. The default file is located at: ***$QNX_DEPLOYMENT_WORKSPACE*/infra/product/AnM/boards/*platform*.ext/mksysimage/*platform*-mksysimage.cfg**

You can also specify your own file by using the `-c` option in `mksysimage.py`. Setting this option will enable you to further customize your tar files and images. For more information about changing the `mksysimage.py` configuration, see "Configuration file for `mksysimage.py`" in *Getting Started*. If you want `mksysimage.py` to skip some steps of the image-generation process and generate only certain types of intermediate files, see "Troubleshooting tips" in the same guide.

Examples:

To run `mksysimage.py`, you need to specify the platform, its variant, and the output path.

The following example reads the default configuration file for the **omap5uevm** platform and creates three images and their corresponding **.tar** files in the specified output path called **/tmp**:

```
$QNX_DEPLOYMENT_WORKSPACE/infra/utils/scripts/mksysimage.py
 -o /tmp/ omap5uevm.external
```

Exit status:

0

The specified image file was created successfully.

>0

An error occurred.

Caveats:

None.

mktar.py

*Create a **.tar** file containing a filesystem for a specified board variant. The mksysimage.py (wrapped by `mksysimage.sh` (Linux) or `mksysimage.bat` (Windows)) calls this command. You shouldn't call this command on its own.*

Syntax:

```
mktar.py  [-o output] [options] ... [board_name_argument]
```

Runs on:

Windows, Linux. Must be run using Python 2.7.5.

Options:

`--bars`

Include new-style (BAR) applications (implies `--no-defaults`)

`--compress`

Use the given compression method: `auto (default)`, `none`, `gzip`, or `bzip2`.

`--cpu` ***CPUDIR***

Set the system architecture (e.g., **\"armle-v7\"**).

`-f` ***SETNAME***, **`--fileset=`*** ***SETNAME***

Include the specified fileset.

`--help`

Display a help message showing `mktar` usage information.

`--no-defaults`

Don't include the board's default filesets or applications.

`-o` ***OUTPUT***, **`--output=`*** ***OUTPUT***

Write to the specified output file.

`-p, --package`

Include the given package (implies `--no-defaults`).

`--prefix` ***PREFIX***

Prefix each path with the specified string.

`--profile`

Specify the profile XML file. If it's not specified, the file name is **profile.xml**.

-P *PRODUCT_BOARDNAME*, --product=*PRODUCT_BOARDNAME*

Specify the name of the product and the board. The entries that can be used for *productname* are the names of the subdirectories found in ***$QNX_DEPLOYMENT_WORKSPACE*/target/product/** and the possible entries for *boardname* are found in ***$QNX_DEPLOYMENT_WORKSPACE*/target/product/*productname*/boards/**.

You should refer to the *Getting Started* guide for the value of the *QNX_DEPLOYMENT_WORKSPACE* variable.

-Q QT_PATH, --qt=QT_PATH

Specify the location where Qt is installed on the host computer. You can set the *QNX_QT* environment variable to specify the default location to use when this option isn't specified.

-s, --symbols

Search in the **runtime-symbols/** folder for the unstripped binaries.

-v, --verbose

Increase verbosity.

-z

Compress with `gzip`.

Description:

The `mktar.py` utility creates a **.tar** file containing the filesets listed in the board-specific profile, resulting in proper ownership and permissions. Don't use the `mktar.py` script on its own. Instead, use the *`mksysimage.py`* (p. 53) script, which calls the `mktar.py` script. To create a **.tar** file with no image, you can call the *`mksysimage.py`* (p. 53) script with the `--no-mkimage` option to create **.tar** files and no image.

If you use `mktar.py` without specifying the profile option `--profile`, then by default, the `mktar` utility will use the file **profile.xml** located in **/boards/*board*.external**.

The `mktar` script uses Python's `tarfile` module to generate a **.tar** file with the appropriate permissions. The list of included files is determined by a board profile, which is stored as **profile.xml** in the **board** directory, whose path is:

***$QNX_DEPLOYMENT_WORKSPACE*/target/boards/omap5uevm/profile.xml**.

For more information and for details about dependencies, see the file located at:

***$QNX_DEPLOYMENT_WORKSPACE*/infra/utils/pymodules/qnx/config.py**

You can overwrite the default profile by using the `--profile` option. The `-vvv` option shows the search path used to find files. The typical search path is implemented in the file ***$QNX_DEPLOYMENT_WORKSPACE*/infra/utils/pymodules/qnx/path.py:get_stage_locator**.

For more information about how the paths are searched, see "Understanding search paths" in *Getting Started*.

Exit status:

0

A **.tar** file was generated successfully.

>0

An error occurred.

Caveats:

None.

Chapter 7
Keyboard (`keyboard-imf`)

Display and manage the on-screen keyboard

Syntax:

```
keyboard-imf [-d device] [-U group:user]
```

Runs on:

QNX Neutrino

Options:

`-d`

The display required for the board (see "*Display types* (p. 61)" below).

`-U` *UID*`:` *GID*

The user ID and the group ID under which to run `keyboard-imf`.

Description:

The `keyboard-imf` service acts as an intermediary between keyboard-dependent applications and underlying keyboard services. For instance, the keyboard service works with the keyboard provided by the HMI to display and manage the on-screen keyboard, or with a physical keyboard to enable input from that keyboard.

The keyboard service lets applications communicate with the on-screen keyboard through PPS objects. It allows them to:

- show and hide the keyboard
- know the keyboard height (in pixels) so they can, if necessary, adjust their displays to fit into the remaining available screen area
- accept text entries and know how many characters have been entered

Interaction of HMI, keyboards, and applications

The diagram below shows how `keyboard-imf` interacts with the HMI virtual keyboards:

- The HMI virtual keyboards (HTML inside the Navigator, or Qt standalone) create the **/pps/system/keyboard/control** and **/pps/system/keyboard/status** PPS objects, and publish their activities to them.
- The `keyboard-imf` service subscribes to these objects to be able to receive information from the HMI.
- The `keyboard-imf` service creates the **/pps/services/input/control** PPS object, to which it publishes information received from the HMI and the applications. This object is for internal communication between `keyboard-imf` and the HMI; other components and applications don't need to publish or subscribe to this object.

- Applications, such as Weblauncher and Qt runtime, subscribe to **/pps/services/input/control** to learn about keyboard presence, height, etc., and publish information, such as the user input and the number of characters entered.

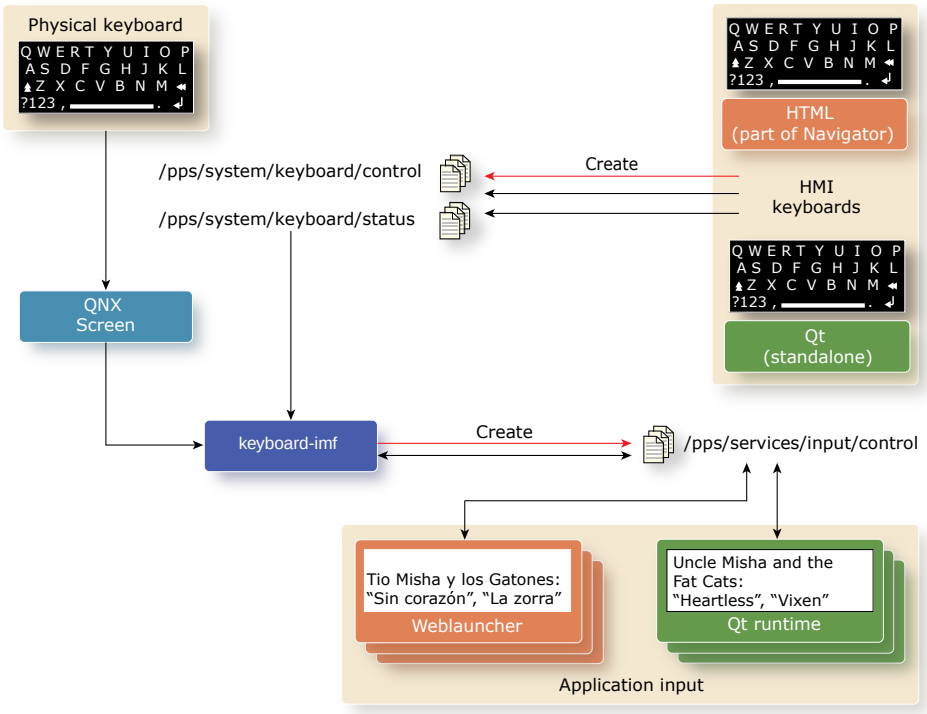

Figure 2: Keyboards, `keyboard-imf` and applications.

Physical keyboard

To use a physical keyboard (connected through a USB port), you need to:

- Configure the `globals` *input* parameter in the Screen configuration file (**graphics.conf**) to accept input from a physical keyboard. For more information, see "Configuration parameters for **globals**" in the *Screen Graphics Subsystem Developer's Guide*.
- Make sure that your system has the language-specific key-mapping files for the languages you will support. These files should be in the **/usr/share/keyboard/** directory.
- Set the *HAS_SOFTWARE_KEYBOARD* environment variable to true or false on the target. You can use the variable to change your application's behavior. For example, the Browser app resizes to move the **URL bar** above the software keyboard if the environment variable is set to true. Conversely, if the variable is set to false, the Browser app doesn't resize.

  You can set the environment variable in the **/scripts/env.override.variant** file.

For more information about how `keyboard-imf` interacts with applications and underlying keyboard services, see "*Interaction of HMI, keyboards, and applications* (p. 59)".

Display types:

The -d option must be set for your board's display. For OMAP5432 boards, set it to **hdmi**; for SABRE Smart Device boards, set it to **internal**. Possible display types are listed in the **/etc/system/config/graphics.conf** configuration file. When you are configuring your system, you will need to edit this file and enter the display type(s) supported on your board, as well as other graphics configuration values.

For more information about the **/etc/system/config/graphics.conf** configuration file and how to configure it, see the chapter "Screen Configuration" in the *Screen Graphics Subsystem Developer's Guide*.

PPS objects:

The HMI keyboard service creates these PPS objects to communicate with keyboard-imf:

- **/pps/system/keyboard/control**
- **/pps/system/keyboard/status**

In addition, the keyboard-imf service creates this PPS object:

- **/pps/services/input/control/**

This object is for internal communication between keyboard-imf and the HMI; other components and applications don't need to publish or subscribe to this object.

Chapter 8
Network Manager (`net_pps`)

PPS interface to the network manager

Syntax:

```
net_pps [-A addr:port] [-a]
        [-c file] [-d] [if0...] [-m]
        [-P script] [-p prefix] [-r name [if0...]]
        [-S uid] [-s] [-u] &
```

Runs on:

QNX Neutrino

Options:

`-A` *addr*`:`*port*

Proxy to publish to when proxy authentication is required.

`-a`

Automatically configure any discovered interfaces when the link state indicates that the interfaces are connected.

`-d`

Enable debug messages (to **stdout**).

`if0...`

Specify prioritized list of interfaces to be considered for multihomed operation, preference of default routes, **confstr** resolver configuration, etc.

`-m`

Use multipath routes.

`-P` *script*

Specify the script to run for updating proxy settings.

`-p` *prefix*

Add this prefix to all executable paths. This has no effect without the `-s` option.

`-r` *name* `[if0...]`

Create another routing domain (called *name*) with the following prioritized interface list.

`-S` *uid*

Run subprocesses as this *uid*.

-s

Use standard filepaths for subprocess executables (defaults: **system=/usr/sbin/ user=/usr/bin/**).

-u

Automatically assume the interface is connected based on its *up* state. This allows shim drivers as well as drivers that don't issue link state changes to work.

Description:

The net_pps service offers a PPS interface for communicating with the QNX network manager. For more information about QNX support for networking, see "Networking Architecture" in the *System Architecture Guide*.

PPS objects:

For more information about the instructions net_pps can send to the network manager and the information it can receive, see the relevant PPS object and directory descriptions:

- **/pps/services/networking/all/proxy**
- **/pps/services/networking/all/interfaces/**
- **/pps/services/networking/all/status_public**
- **/pps/services/networking/control**

Chapter 9
Shutdown (coreServices2)

Provide access through Persistent Publish/Subscribe (PPS) communication to a variety of services, including shutdown

Syntax:

```
coreServices2 [-C configuration file] [-d] [-l none | module [,module]*]
              [-M UID:GIG] [-r path] [-S UID:GIG] [-U UID:GIG] [-v]*
```

Runs on:

QNX Neutrino

Options:

-C *filename*

The filepath and filename of the configuration file.

-d

Run in foreground instead of as a daemon (default).

-l none | *module* [, *module*]*

If specified, use this list of dynamic modules instead of the dynamic modules listed in the configuration file.

-M

The username or the *UID:GID,GID* specifying the user and group IDs of the monitor process.

-r *path*

Specify the root path to the PPS service. The default is **/pps/services/**.

-S

The username or the *UID:GID,GID* specifying the user and group IDs of the spawner process.

-U

The username or the *UID:GID,GID* specifying the user and group IDs of the main server process.

-v

Set verbosity of output to `sloginfo`.

Description:

The coreServices2 utility provides a single point from which to ask the system to run a variety of services. It manages the operations involved in communicating with multiple services using PPS objects. This design means that the requesting component or application only needs to publish and subscribe to the relevant PPS objects.

The QNX SDK for Apps and Media uses coreServices2 to provide access from the HMI to the shutdown service.

coreServices2 objects:

The coreServices2 service maintains a separate object for each service to which it gives access. An object may be static (compiled into the coreServices2 binary) or dynamic (loaded through *dlopen()* at runtime).

Most basic services are static, but some services that are large (or that require large shared libraries) and are not needed by all implementations are made available as dynamic modules. The coreServices2 service doesn't use any dynamically loaded modules and maintains the following statically loaded object:

shutdown

Shut down and reboot the system (see shutdown in the *Utilities Reference*).

PPS objects:

The coreServices2 service publishes or subscribes to the following PPS object:

- **/pps/services/system/info**

Configuration file:

A configuration file specifies the core services that can be used. The name and location of the file is specified by the -C option. In QNX Apps and Media targets, the configuration file is located at **/etc/system/config/coreServices2.json**.

The file has the following syntax:

```
{
"static_modules" : comma-separated string of static module names
"dynamic_modules" : comma-separated string of dynamic module names
"disable_procmon" : [true|false]
"disable_hwid" : [true|false]
}
```

For example, the configuration file on the reference image includes only the shutdown service, which is a static module:

```
{
"static_modules" : "shutdown",
"dynamic_modules" : "",
"disable_procmon" : true,
```

```
"disable_hwid" : true
}
```

Chapter 10
System Launch and Monitor (SLM)

Automate process management

Syntax:

`slm` *configuration_file*

Runs on:

QNX Neutrino

Description:

System Launch and Monitor (SLM) is started early in the boot sequence to launch complex applications consisting of multiple processes that must be started in a specific order.

SLM is a utility controlled by a configuration file. The configuration file specifies any interprocess dependencies, the processes to run, and the process properties. For example, suppose a multimedia application needs the services of the audio subsystem and the database server, which in turn requires the Persistent Publish/Subscribe (PPS) service. When SLM learns of these one-way dependencies when reading the configuration file, the service internally constructs a *directed acyclic graph (DAG)*. The DAG represents the workflow of the underlying processes and is sorted to produce a partial ordering for scheduling the processes so that all control-flow dependencies are respected. In this example, SLM would first verify that PPS is running before starting the database server, and then check that the database server is running before starting the multimedia application.

For more information about how to use SLM, see `slm` in the *Utilities Reference*.

Index