x4531 - Multi Platform

Version 1

Multi Platform

[0] **getFeatureInfos**() → capabilityMask, numPlatforms, numPlatformDescr, numHosts, currentHost
[1] **getPlatformDescriptor**(platformDescriptorIndex) → platformDescriptor
[2] **getHostPlatform**(hostIndex) → hostIndex, status, platform, platformSource, autoPlatform, autoDescr
[3] **setHostPlatform**(hostIndex, platformIndex) → void
[event0] **PlatformChange** → hostIndex, platformIndex, platformSrc

Platform Descriptors Examples

Overview

Multi-Platform devices behave specifically to the host operating system and version, described as platforms. x4531 feature reports the Platforms defined and supported by the device, and which Platform is configured per host. It does not describe how a Platform affects the device.

A Platform, identified with a unique number, is a collection of OS version ranges, and can cover one OS range("iOS 8") or multiple ranges or multiple OSes ("MacOs all versions + iOs 1 to 7"). Each range is defined by a Platform Descriptor: an OS name bitfield with one version range (all versions if the range is 0.0..0.0). See Platform Descriptor and Platform Descriptors Examples

The Platform selected when the device connects to a host is persistent. It can be determined automatically at pairing or connection if the device supports automatic OS detection, and set by the user or software (then should not be overwritten by OS detection).

Functions and Events

[0] getFeatureInfos() → capabilityMask, numPlatforms, numPlatformDescr, numHosts, currentHost

Get the number of Platforms and Platform Descriptors defined by the device, the number of host configurations supported by the device and the index of the currently active host (logically the caller). Also returns the device's capability mask for.

Parameters

none

Return

[2bytes] capabilityMask

Sub-features implemented by the device. A bit is 1 if associated capability is available, else it has to be 0.

- **osDetection**: automatic osDetection is implemented.
- **setHostPlatform**: setHostPlatform is supported.

[1byte] numPlatforms

Number of Platforms defined by the device.

[1byte] numPlatformDescr

Number of Platform Descriptors defined by the device.

[1byte] numHosts

Number of hosts / channels.

[1byte] currentHost

Current host index [0..numHosts - 1].

[1byte] currentHostPlatform

Current host’s Platform index [0..numPlatforms - 1].

Table 1. getFeatureInfos() response packet format.

byte \ bit	7	6	5	4	3	2	1	0
0	-	-	-	-	-	-	setHostPlatform	osDetection
1	-	-	-	-	-	-	-	-
2	numPlatforms							
3	numPlatformDescr							
4	numHosts							
5	currentHost							
6	currentHostPlatform							
7..15	reserved							

Errors

none

[1] getPlatformDescriptor(platformDescriptorIndex) → platformDescriptor

Get the Platform Descriptor specified by index.

Parameters

[1byte] platformDescriptorIndex

Index of the Platform Descriptor [0..numPlatformDescr-1] (returned by getFeatureInfos()).

Table 2. getPlatformDescriptor() request packet format.

byte \ bit	7	6	5	4	3	2	1	0
0	platformDescriptorIndex							
1..15	reserved							

Return

getPlatformDescriptor() returns a Platform Descriptor structure.

Platform Descriptor

A Platform is the union of one or more Platform Descriptors. Each Platform Descriptor contains the Platform index it belongs to, an OS list - <osMask> bit field - and a <fromVersion>.<fromRevision>..<toVersion>.<toRevision> range. If all version and revision fields are 0, then all versions for the OS list are covered.

See Platform Descriptors Examples for illustrated examples.

[1byte] platformIndex

Index of the Platform [0..numPlatforms-1].

[1byte] platformDescriptorIndex

Index of the Platform Descriptor [0..numPlatformDescr-1].

[2bytes] osMask

Bitfield of OS covered by the Platform Descriptor.

[1byte] fromVersion

OS Version start number, 0 if no lower limit.

[1byte] fromRevision

OS Revision start number, 0 if not lower limit.

[1byte] toVersion

OS Version end number, 0 if no upper limit.

[1byte] toRevision

OS Revision end number, 0 if no upper limit.

Table 3. getPlatformDescriptor() response packet format.

byte \ bit	7	6	5	4	3	2	1	0
0	Platform index							
1	Platform Descriptor index							
2	WebOS	iOS	MacOS	Android	Chrome	Linux	WinEmb	Windows
3	-	-	-	-	-	-	-	Tizen
4	From Version							
5	From Revision							
6	To Version							
7	To Revision							
8..15	reserved							

Errors

(2) InvalidArgument

Invalid index.

[2] getHostPlatform(hostIndex) → hostIndex, status, platform, platformSource, autoPlatform, autoDescr

Get the Platform index configured for a specific host.

Parameters

[1byte] hostIndex

Channel / host index. 0xFF = Current Host.

Table 4. getHostPlatform() request packet format.

byte \ bit	7	6	5	4	3	2	1	0
0	Host index							
1..15	reserved							

Return

[1byte] hostIndex

Same value as the parameter.

[1byte] status

Pairing status (0=empty slot, 1=paired)

[1byte] platformIndex

Platform index currently used (auto or manually set).

- 0xFF - Undefined (slot empty).
- else - Platform index.

[1byte] platformSource

Origin of current Platform index configuration:

- 0 - Default, current Platform index initialized with default value.
- 1 - Auto, current Platform index has been set by auto-detection.
- 2 - Manual, current Platform index has been set manually (user).
- 3 - Software, current Platform index has been set by software (user).

[1byte] autoPlatform

Platform index automatically defined at pairing.

- 0xFF - Undefined / failed.
- else - Platform index.

[1byte] autoDescr

Platform Descriptor index automatically defined at pairing.

- 0xFF - Undefined / failed.
- else - PlatformDescr index.

Table 5. getHostPlatform() response packet format.

byte \ bit	7	6	5	4	3	2	1	0
0	Host index							
1	Status							
2	Platform index							
3	Platform source							
4	Auto platform							
5	Auto descriptor							
6..15	reserved							

Errors

(2) InvalidArgument

Invalid hostIndex.

[3] setHostPlatform(hostIndex, platformIndex) → void

Change the Platform index configuration for a specific host.

Parameters

[1byte] hostIndex

Channel / host index. 0xFF = Current Host.

[1byte] platformIndex

Platform index to set.

Table 6. setHostPlatform() request packet format.

byte \ bit	7	6	5	4	3	2	1	0
0	Host index							
1	Platform index							
2..15	reserved							

Return

none

Errors

(2) InvalidArgument

Invalid hostIndex or platformIndex.

(5) NotAllowed

Operation not permitted/not implemented.

[event0] PlatformChange → hostIndex, platformIndex, platformSrc

Event sent when current host’s Platform index is changed by the user (or other mean than software receiving the event).

Values

[1byte] hostIndex

Channel / host index.

[1byte] platformIndex

New host Platform index setting.

[1byte] platformSource

Source of the new host Platform index setting.

Table 7. PlatformChange() event packet format.

byte \ bit	7	6	5	4	3	2	1	0
0	Host index							
1	Platform index							
2	Platform source							
3..15	reserved							

Platform Descriptors Examples

[[Platform Descriptors Examples]] How to translate a Platform into Platform Descriptor ? Let us take an example with a device implementing 3 different behaviours, and thus defining 3 Platforms:

```
Platform[0]: Applies to all versions of Windows + Linux + Android.
Platform[1]: Applies to all versions of MacOs and iOS up to v2.4.
Platform[2]: Applies to iOS from v2.5.
```

Platform[0] covers all versions of Windows, Linux and Android. We could define a Platform Descriptor foe each OS, but, given they have the same version range (0.0..0.0), they can be merged into 1 Platform Descripor:

```
platformDescr[0] = platform=0, osMask=Windows + Linux + Android,
                   fromVersion=0, fromRevision=0, toVersion=0, toRevision=0
```

Platform[1] covers 2 OSes with different version ranges, thus requires 2 Platform Descriptors:

```
platformDescr[1] = platform=1, osMask=MacOs,
                   fromVersion=0, fromRevision=0, toVersion=0, toRevision=0
platformDescr[2] = platform=1, osMask=IOS,
                   fromVersion=0, fromRevision=0, toVersion=2, toRevision=4
```

Platform[2] covers 1 OS described with 1 Platform Descriptor:

```
platformDescr[3]: platform=2, osMask=IOS,
                  fromVersion=2, fromRevision=5, toVersion=0, toRevision=0
```

So in final the device declares 3 Platforms with 4 Platform Descriptors:

Table 8. getFeatureInfos() response example (compacted).

Capability Mask	Num Platforms	Num Platform Descr	Num Hosts	Current Host	Current Host Platform
3	3	4	4	1	1

Table 9. getPlatformDescriptor(0)..(3) responses example (compacted).

Platform Decriptor index	Platform index	OS Mask	From Version	From Revision	To Version	To Revision
0	0	Windows + Linux + Android	0	0	0	0
1	1	MacOs	0	0	0	0
2	1	iOS	0	0	2	4
3	2	iOS	2	5	0	0

ChangeLog

- Version 0: Initial version.
- Version 1: add WebOS and Tizen OS platforms