

# OpenCable Application Platform Specifications OCAP Extensions

## OCAP Front Panel Extension

### OC-SP-OCAP-FPEXT-I04-100603

**ISSUED**

#### **Notice**

This OpenCable specification is the result of a cooperative effort undertaken at the direction of Cable Television Laboratories, Inc. for the benefit of the cable industry and its customers. This document may contain references to other documents not owned or controlled by CableLabs. Use and understanding of this document may require access to such other documents. Designing, manufacturing, distributing, using, selling, or servicing products, or providing services, based on this document may require intellectual property licenses from third parties for technology referenced in this document.

Neither CableLabs nor any member company is responsible to any party for any liability of any nature whatsoever resulting from or arising out of use or reliance upon this document, or any document referenced herein. This document is furnished on an "AS IS" basis and neither CableLabs nor its members provides any representation or warranty, express or implied, regarding the accuracy, completeness, noninfringement, or fitness for a particular purpose of this document, or any document referenced herein.

© Copyright 2005-2010 Cable Television Laboratories, Inc.  
All rights reserved.

## Document Status Sheet

<b>Document Control Number:</b>	OC-SP-OCAP-FPEXT-I04-100603			
<b>Document Title:</b>	OCAP Front Panel Extension			
<b>Revision History:</b>	I01 – Released 6/24/05			
	I02 – Released 12/20/07			
	I03 – Released 6/12/09			
	I04 – Released 6/3/10			
<b>Date:</b>	June 3, 2010			
<b>Status:</b>	<del>Work in Progress</del>	<del>Draft</del>	<b>Issued</b>	<del>Closed</del>
<b>Distribution Restrictions:</b>	<del>Author Only</del>	<del>CL/Member</del>	<del>CL/Member/ Vendor</del>	<b>Public</b>

### Key to Document Status Codes

- Work in Progress** An incomplete document, designed to guide discussion and generate feedback that may include several alternative requirements for consideration.
- Draft** A document in specification format considered largely complete, but lacking review by Members and vendors. Drafts are susceptible to substantial change during the review process.
- Issued** A stable document, which has undergone rigorous member and vendor review and is suitable for product design and development, cross-vendor interoperability, and for certification testing.
- Closed** A static document, reviewed, tested, validated, and closed to further engineering change requests to the specification through CableLabs.

### Trademarks

CableLabs<sup>®</sup>, DOCSIS<sup>®</sup>, EuroDOCSIS<sup>™</sup>, eDOCSIS<sup>™</sup>, M-CMTS<sup>™</sup>, PacketCable<sup>™</sup>, EuroPacketCable<sup>™</sup>, PCMM<sup>™</sup>, CableHome<sup>®</sup>, CableOffice<sup>™</sup>, OpenCable<sup>™</sup>, OCAP<sup>™</sup>, CableCARD<sup>™</sup>, M-Card<sup>™</sup>, DCAS<sup>™</sup>, tru2way<sup>™</sup>, and CablePC<sup>™</sup> are trademarks of Cable Television Laboratories, Inc.

# Contents

<b>1</b>	<b>SCOPE</b> .....	<b>1</b>
1.1	OCAP FP PURPOSE .....	1
1.2	OCAP FP REQUIREMENTS .....	1
1.3	CONVENTIONS .....	1
1.4	FRONT PANEL SPECIFIC REQUIREMENTS .....	2
1.4.1	<i>Specification Language</i> .....	2
<b>2</b>	<b>REFERENCES</b> .....	<b>3</b>
2.1	DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE.....	3
2.2	NORMATIVE REFERENCES .....	3
2.3	INFORMATIVE REFERENCES .....	3
2.4	REFERENCE ACQUISITION.....	4
<b>3</b>	<b>TERMS AND DEFINITIONS</b> .....	<b>5</b>
<b>4</b>	<b>ABBREVIATIONS AND ACRONYMS</b> .....	<b>6</b>
<b>5</b>	<b>FRONT PANEL</b> .....	<b>7</b>
5.1	INTRODUCTION.....	7
5.1.1	<i>OpenCable Host Front Panel Example (Informative)</i> .....	7
5.2	MODES OF DISPLAY .....	7
5.3	INDICATORS.....	7
5.4	SECURITY .....	8
5.5	API SUPPORT PROPERTY .....	8
<b>6</b>	<b>REGISTRY OF CONSTANTS</b> .....	<b>9</b>
6.1	OCAP-SPECIFIC JAVA CONSTANTS .....	9
<b>ANNEX A</b>	<b>FRONT PANEL API</b> .....	<b>10</b>
	PACKAGE ORG.OCAP.HARDWARE.FRONTPANEL.....	10
	ORG.OCAP.HARDWARE.FRONTPANEL INTERFACE BLINKSPEC.....	11
	<i>getIterations</i> .....	11
	<i>getMaxCycleRate</i> .....	11
	<i>setIterations</i> .....	11
	<i>getOnDuration</i> .....	12
	<i>setOnDuration</i> .....	12
	ORG.OCAP.HARDWARE.FRONTPANEL INTERFACE BRIGHTSPEC .....	13
	<i>OFF</i> .....	13
	<i>getBrightness</i> .....	13
	<i>getBrightnessLevels</i> .....	13
	<i>setBrightness</i> .....	14
	ORG.OCAP.HARDWARE.FRONTPANEL INTERFACE COLORSPEC .....	15
	<i>BLUE</i> .....	15
	<i>GREEN</i> .....	15
	<i>RED</i> .....	15
	<i>YELLOW</i> .....	16
	<i>ORANGE</i> .....	16
	<i>getColor</i> .....	16
	<i>getSupportedColors</i> .....	16
	<i>setColor</i> .....	16
	ORG.OCAP.HARDWARE.FRONTPANEL CLASS FRONTPANELMANAGER.....	17

<i>FrontPanelManager</i> .....	18
<i>getInstance</i> .....	18
<i>reserveTextDisplay</i> .....	18
<i>reserveIndicator</i> .....	18
<i>releaseTextDisplay</i> .....	18
<i>releaseIndicator</i> .....	19
<i>getTextDisplay</i> .....	19
<i>getSupportedIndicators</i> .....	19
<i>getIndicatorDisplay</i> .....	19
<i>isTextDisplaySupported</i> .....	19
ORG.OCAP.HARDWARE.FRONTANEL INTERFACE INDICATOR .....	20
<i>getBrightSpec</i> .....	20
<i>setBrightSpec</i> .....	20
<i>getColorSpec</i> .....	21
<i>setColorSpec</i> .....	21
<i>getBlinkSpec</i> .....	21
<i>setBlinkSpec</i> .....	21
ORG.OCAP.HARDWARE.FRONTANEL INTERFACE INDICATORDISPLAY .....	22
<i>POWER</i> .....	22
<i>RFBYPASS</i> .....	22
<i>MESSAGE</i> .....	22
<i>RECORD</i> .....	22
<i>REMOTE</i> .....	23
<i>getIndicators</i> .....	23
ORG.OCAP.HARDWARE.FRONTANEL INTERFACE SCROLLSPEC .....	24
<i>getMaxHorizontalIterations</i> .....	24
<i>getMaxVerticalIterations</i> .....	24
<i>getHorizontalIterations</i> .....	25
<i>getVerticalIterations</i> .....	25
<i>setHorizontalIterations</i> .....	25
<i>setVerticalIterations</i> .....	25
<i>getHoldDuration</i> .....	25
<i>setHoldDuration</i> .....	25
ORG.OCAP.HARDWARE.FRONTANEL INTERFACE TEXTDISPLAY .....	27
<i>TWELVE_HOUR_CLOCK</i> .....	28
<i>TWENTYFOUR_HOUR_CLOCK</i> .....	28
<i>STRING_MODE</i> .....	28
<i>getBrightSpec</i> .....	28
<i>getColorSpec</i> .....	28
<i>getBlinkSpec</i> .....	28
<i>getScrollSpec</i> .....	29
<i>getMode</i> .....	29
<i>getNumberColumns</i> .....	29
<i>getNumberRows</i> .....	29
<i>getCharacterSet</i> .....	29
<i>setClockDisplay</i> .....	29
<i>setTextDisplay</i> .....	30
<i>setWrap</i> .....	30
<i>eraseDisplay</i> .....	31
<b>APPENDIX I REVISION HISTORY</b> .....	<b>32</b>

## Figures

Figure 5–1 - 4 Digit, 7-Segment LED Display .....7

## Tables

Table 5–1 - Permissions .....8  
Table 5–2 - API Support Property .....8

This page intentionally left blank.

# 1 SCOPE

This document defines a minimal profile for a Front Panel (FP) software environment for digital cable receivers with local storage. This platform is a modular extension to the OpenCable Application Platform [OCAP]. The OCAP 1.1 Profile, and the OCAP FP Profile, were developed by Cable Television Laboratories, Inc. (CableLabs) in conjunction with representatives from a number of its member cable operating companies, as well as leading software and hardware firms.

The OCAP FP is based on the OCAP 1.1 Profile and includes that document in its entirety.

## 1.1 OCAP FP Purpose

OCAP FP is an application interface that includes all required Application Program Interfaces (APIs), content and data formats, and protocols up to the application level. Applications developed to the OCAP FP Profile will be executed on OpenCable-compliant Host devices. The OCAP FP Profile allows cable operators to deploy their applications and services that use FP on all OpenCable-compliant host devices connected to their networks that contain a Front Panel.

The OCAP FP platform SHALL be applicable to a wide variety of hardware and operating systems that contain a hardware FP, to allow Consumer Electronics (CE) manufacturers flexibility in implementation. A primary objective in defining OCAP FP is to enable competing implementations of the OCAP FP platform by CE manufacturers.

## 1.2 OCAP FP Requirements

The OCAP FP platform has been designed to meet specific requirements that are not commonly applied to other FP environments. Some of these requirements are related to customer communication obligations that cable operators face, such that FP event descriptions might be maintained by applications, and that the platform supports FP applications deployed by different service providers that have no *a priori* knowledge of each other and may compete for resources. Specific requirements include:

- A front panel API SHALL be provided that allows access to and modification of what's being displayed on a front panel when a front panel is supported by the hardware.
- The front panel API SHALL be agnostic as to the type of front panel hardware (e.g., 7 segment LED, LCD).
- The front panel API SHALL include the capability to access and modify indicators for standardized indications including power, message, RF bypass, and record.

## 1.3 Conventions

The following conventions are used in this manual:

- The following font type is used to indicate code examples, names of properties, and other information that MUST be entered exactly as-is: `code example font`
- **Boldfaced** text is used as emphasis.

## 1.4 Front Panel Specific Requirements

### 1.4.1 Specification Language

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

“SHALL”	This word means that the item is an absolute requirement of this specification.
“SHALL NOT”	This phrase means that the item is an absolute prohibition of this specification.
“SHOULD”	This word means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
“SHOULD NOT”	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
“MAY”	This word means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

## 2 REFERENCES

This section provides the normative and informative references used to create this specification.

### 2.1 DVB-GEM and DVB-MHP Specification Correspondence

This specification contains requirements specific to the OCAP Front Panel Extension and does not correspond to any DVB-GEM 1.0.2 section and is not contained within DVB-MHP 1.0.3.

### 2.2 Normative References

**Note:** Information contained in these normative references is required for all implementations. Notwithstanding, intellectual property rights may be required to use or implement these normative references.

In order to claim compliance with this specification, it is necessary to conform to the following standards and other works as indicated, in addition to the other requirements of this specification. Notwithstanding, intellectual property rights may be required to use or implement such normative references.

The following documents contain provisions which, through reference in this text, constitute provisions of the present documents. References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific:

- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Some known errata in these references are identified in Annex A of [DVB-MHP 1.0.3]. These errata take precedence over the published reference.

References	Edition	Description
[OCAP]	1.1	OpenCable Application Platform, Profile 1.1, OC-SP-OCAP1.1.3-100603, June 3, 2010, Cable Television Laboratories, Inc.

### 2.3 Informative References

This specification uses the following informative references.

Reference	Edition	Description
[DVB-GEM 1.0.2]	ETSI TS 102 819 v1.3.1 October 2005	Digital Video Broadcasting (DVB); Globally Executable MHP version 1.0.2
[DVB-MHP 1.0.3]	ETSI TS 101 812 v1.3.1 June 2003	Digital Video Broadcasting (DVB);Multimedia Home Platform (MHP) Specification 1.0.3

## 2.4 Reference Acquisition

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027; Phone +1-303-661-9100; Fax +1-303-661-9199; <http://www.cablelabs.com>
- European Telecommunications Standards Institute (ETSI), [www.etsi.org](http://www.etsi.org)

### 3 TERMS AND DEFINITIONS

This specification uses the following terms:

**Front Panel** A display and key input mechanism that is separate from the main display and remote control. Examples of a front panel display include adjacent multi-segmented LEDs or an LCD.

## 4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations:

<b>DTD</b>	Document Type Definition
<b>FP</b>	Front Panel
<b>PRF</b>	Permission Request File

## 5 FRONT PANEL

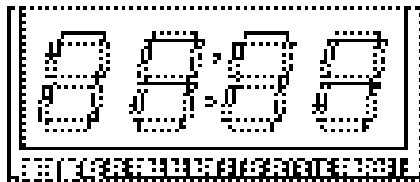
### 5.1 Introduction

Host devices MAY incorporate a front panel display that is separate from a main video display. Examples of a front panel display include multiple adjacent segmented LEDs (light emitting diodes), or small LCD displays. When a Host device incorporates such a display, it SHALL implement the front panel APIs specified in Annex A. When a Host device does not incorporate such a display, the `org.ocap.hardware.frontpanel.FrontPanelManager` class SHALL NOT be present in the implementation. Applications wishing to access a front panel should verify the `FrontPanelManager` class is available. Once an application has determined a front panel is part of the Host device hardware, it can get the front panel manager singleton using the `FrontPanelManager.getInstance` method. To modify a front panel display, an application must first reserve the front panel using the `FrontPanelManager` method.

#### 5.1.1 OpenCable Host Front Panel Example (Informative)

For example, a Host is designed with a front panel display that incorporates a POWER LED and MESSAGE LED, and four (4) 7-segment LED displays in a format, such that time may be displayed, and includes a colon in the middle of the display.

The following is an example of a 4 digit, 7-segment display:



*Figure 5–1 - 4 Digit, 7-Segment LED Display*

As another example, a Host is designed with a front panel display and includes RF Bypass functionality and an RF BYPASS LED. This LED is active when the RF Bypass is active.

### 5.2 Modes of Display

If the Host supports text display, the front panel API can place a front panel in one of two modes: text and clock. In text mode, it is the responsibility of the application to update the display. Along with setting the text, an application can control effects such as blink, scroll, and wrap.

In clock mode, the front panel will display either a 12 or 24 hour clock, settable by the controlling application. The implementation SHALL maintain the clock display until the mode is changed to text.

### 5.3 Indicators

When a Host device supports a front panel display, the front panel API SHALL be able to control all supported indicator lamps, typically LEDs. An application SHALL be able to turn these lamps on and off. The ability to

change color and brightness support is optional. The recommended minimum set of indicators is power, RF bypass, message, and record (if the device supports DVR).

## 5.4 Security

In order to call the `FrontPanelManager.getInstance` method, the calling application must have `MonitorAppPermission("frontpanel")`. In Host devices that contain front panel hardware, the `MonitorAppPermission` javadoc SHALL contain an additional row in the permissions table as follows:

**Table 5–1 - Permissions**

frontpanel	Allows use of the front panel API.	Allows an application to get the front panel manager singleton and use the front panel API to modify the front panel display.
------------	------------------------------------	---

In addition, the enumerated token value type of the `name` attribute of the `ocap:monitorapplication` element type defined in [OCAP] Section 14.2.2.1.1 SHALL be considered to contain the “frontpanel” value.

## 5.5 API Support Property

OCAP Hosts that support the OCAP Front Panel extension SHALL indicate this support with the system properties as defined in section 13.3.14.3 of [OCAP] per Table 5–2.

**Table 5–2 - API Support Property**

Property	Description	Value	Application Access
ocap.api.option.fp	System property indicating that OCAP FP extension is supported by the Host device	“1.0”	signed and unsigned

## 6 REGISTRY OF CONSTANTS

This section itemizes the system constants required by the OpenCable Application Platform Front Panel Extension. This section specifies the values of the public static symbols from various Java APIs.

### 6.1 OCAP-Specific JAVA Constants

All constants are public final. These OCAP 1.1 Profile FPEXT-specific JAVA constants are set in the following packages:

<b>org.ocap.hardware.frontpanel.BrightSpec</b>			
public static final int	OFF		0
<b>org.ocap.hardware.frontpanel.ColorSpec</b>			
public static final byte	BLUE		1
public static final byte	GREEN		2
public static final byte	ORANGE		16
public static final byte	RED		4
public static final byte	YELLOW		8
<b>org.ocap.hardware.frontpanel.IndicatorDisplay</b>			
public static final java.lang.String	MESSAGE		"message"
public static final java.lang.String	POWER		"power"
public static final java.lang.String	RECORD		"record"
public static final java.lang.String	REMOTE		"remote"
public static final java.lang.String	RFBYPASS		"rfbypass"
<b>org.ocap.hardware.frontpanel.TextDisplay</b>			
public static final byte	STRING_MODE		2
public static final byte	TWELVE_HOUR_CLOCK		0
public static final byte	TWENTYFOUR_HOUR_CLOCK		1

## Annex A Front Panel API

### Package org.ocap.hardware.frontpanel

Interface Summary	
<b>BlinkSpec</b>	This interface represents the front panel display blinking specification.
<b>BrightSpec</b>	This interface represents the front panel display brightness specification.
<b>ColorSpec</b>	This interface represents the front panel display Color specification.
<b>Indicator</b>	This interface represents an indicator in the front panel display and allows its properties to be checked and set.
<b>IndicatorDisplay</b>	This interface represents the set of individual indicators on a front panel (e.g. power, recording).
<b>ScrollSpec</b>	This interface represents the scrolling specification of a front panel text display.
<b>TextDisplay</b>	This interface represents one line of characters in a front panel display.

Class Summary	
<b>FrontPanelManager</b>	This class represents an optional front panel display and SHOULD not be present in any device that does not support one.

## org.ocap.hardware.frontpanel Interface BlinkSpec

```
public interface BlinkSpec
```

This interface represents the front panel display blinking specification. If `BlinkSpec` is for `TextDisplay`, all characters blink at the same time.

### Method Summary

int	<b>getIterations()</b> Gets the number of times per minute the display will blink.
int	<b>getMaxCycleRate()</b> Gets the maximum number of times per minute all segments in an LED can blink.
int	<b>getOnDuration()</b> Gets the percentage of time the text will be on during one blink iteration.
void	<b>setIterations(int iterations)</b> Sets the number of times per minute data will blink across all of the LEDs.
void	<b>setOnDuration(int duration)</b> Sets the percentage of time the text display will remain on during one blink iteration.

### Method Detail

#### getIterations

```
int getIterations()  
    Gets the number of times per minute the display will blink.  
    Returns:  
    Number of blink iterations per minute.
```

#### getMaxCycleRate

```
int getMaxCycleRate()  
    Gets the maximum number of times per minute all segments in an LED can blink.  
    Returns:  
    Maximum number of blink iterations per minute. Returns zero if blinking is not supported.
```

#### setIterations

```
void setIterations(int iterations)  
    Sets the number of times per minute data will blink across all of the LEDs.  
    Parameters:  
    iterations - Number of blink iterations per minute.  
    Throws:  
    java.lang.IllegalArgumentException - if the iteration is negative or cannot be supported by  
    the front panel.
```

**getOnDuration**

```
int getOnDuration()
```

Gets the percentage of time the text will be on during one blink iteration.

**Returns:**

Text display blink on percentage duration.

**setOnDuration**

```
void setOnDuration(int duration)
```

Sets the percentage of time the text display will remain on during one blink iteration.

**Parameters:**

`duration` - Text display blink on percentage duration. Setting this value to 100 sets the display no solid, no blinking. Setting this value to 0 effectively turns off the front panel display. Subtracting this value from 100 yields the percentage of off time during one blink iteration.

**Throws:**

`java.lang.IllegalArgumentException` - if the duration is negative or exceeds 100.

## org.ocap.hardware.frontpanel Interface BrightSpec

```
public interface BrightSpec
```

This interface represents the front panel display brightness specification. If the Indicator supports just turn on/off, the brightness value OFF represents turn off and value 1 represents turn on, and a brightness level shall be 2. If the Indicator supports bright/dark/off brightness, the brightness value OFF represents turn off and value 1 represents dark and value 2 represents bright, and a brightness level shall be 3. The brightness level can be any levels corresponding to the Indicator's capability.

### Field Summary

static int	<b>OFF</b> Brightness OFF setting that represents the Indicator is off.
------------	--

### Method Summary

int	<b>getBrightness()</b> Gets the current brightness of the Indicator.
int	<b>getBrightnessLevels()</b> Gets the number of brightness levels supported.
void	<b>setBrightness(int brightness)</b> Sets the brightness of the indicator.

### Field Detail

#### OFF

```
static final int OFF  
    Brightness OFF setting that represents the Indicator is off.
```

### Method Detail

#### getBrightness

```
int getBrightness()  
    Gets the current brightness of the Indicator. Possible return values shall be an integer that meets:  
    OFF =< brightness =< getBrightnessLevels()-1  
Returns:  
    a current brightness value of Indicator.
```

#### getBrightnessLevels

```
int getBrightnessLevels()  
    Gets the number of brightness levels supported. The minimum support brightness level SHALL be 2, i.e.,  
    two levels that includes OFF and brightness=1 (on). This provides an on/off capability.
```

**Returns:**

Supported indicator brightness levels.

**setBrightness**

void **setBrightness**(int brightness)

throws java.lang.IllegalArgumentException

Sets the brightness of the indicator. Setting the brightness level to OFF turns the indicator off.

**Parameters:**

brightness - Indicator brightness.

**Throws:**

java.lang.IllegalArgumentException - if the brightness value is not an integer that meets:

OFF =< brightness =< getBrightnessLevels()-1

**org.ocap.hardware.frontpanel**  
**Interface ColorSpec**

```
public interface ColorSpec
```

This interface represents the front panel display Color specification.

## Field Summary

static byte	<b>BLUE</b> Indicator color blue.
static byte	<b>GREEN</b> Indicator color green.
static byte	<b>ORANGE</b> Indicator color orange.
static byte	<b>RED</b> Indicator color red.
static byte	<b>YELLOW</b> Indicator color yellow.

## Method Summary

byte	<b>getColor()</b> Gets the current color of the indicator.
byte	<b>getSupportedColors()</b> Gets the supported colors.
void	<b>setColor(byte color)</b> Sets the color of this indicator.

## Field Detail

### BLUE

```
static final byte BLUE  
Indicator color blue.
```

### GREEN

```
static final byte GREEN  
Indicator color green.
```

### RED

```
static final byte RED  
Indicator color red.
```

**YELLOW**

static final byte **YELLOW**  
Indicator color yellow.

**ORANGE**

static final byte **ORANGE**  
Indicator color orange.

## Method Detail

**getColor**

byte **getColor**()  
Gets the current color of the indicator. See definitions of BLUE, GREEN, RED, YELLOW, and ORANGE for possible values.  
**Returns:**  
Indicator color.

**getSupportedColors**

byte **getSupportedColors**()  
Gets the supported colors. The value returned SHALL contain values for the possible color set OR'ed together. BLUE, GREEN, RED, YELLOW, and ORANGE for possible values.  
**Returns:**  
Supported color set.

**setColor**

void **setColor**(byte color)  
throws java.lang.IllegalArgumentException  
Sets the color of this indicator.  
**Parameters:**  
color - Indicator color.  
**Throws:**  
java.lang.IllegalArgumentException - if the value of the color parameter is not in the supported color set.

## org.ocap.hardware.frontpanel Class FrontPanelManager

```
java.lang.Object
└─ org.ocap.hardware.frontpanel.FrontPanelManager
```

```
public class FrontPanelManager
extends java.lang.Object
```

This class represents an optional front panel display and SHOULD not be present in any device that does not support one. A front panel may include a text based display with one or more rows of characters. This API is agnostic as to the type of hardware used in the display (e.g. segmented LED, LCD). The display may also contain individual indicators for status indication such as power.

### Constructor Summary

protected	<b>FrontPanelManager</b> ( ) Protected constructor.
-----------	--

### Method Summary

IndicatorDisplay	<b>getIndicatorDisplay</b> ( java.lang.String[] indicators) Gets the individual indicators display.
static FrontPanelManager	<b>getInstance</b> ( ) Gets the singleton instance of the front panel manager.
java.lang.String[]	<b>getSupportedIndicators</b> ( ) Gets the set of available indicators.
TextDisplay	<b>getTextDisplay</b> ( ) Gets the front panel text display.
boolean	<b>isTextDisplaySupported</b> ( ) Provides an indication of whether or not text display is supported.
void	<b>releaseIndicator</b> ( java.lang.String indicator) Releases a front panel indicator from a previous reservation.
void	<b>releaseTextDisplay</b> ( ) Releases the front panel text display from a previous reservation.
boolean	<b>reserveIndicator</b> (ResourceClient resourceClient, java.lang.String indicator) Reserves one of the indicators for exclusive use by an application.
boolean	<b>reserveTextDisplay</b> (ResourceClient resourceClient) Reserves the front panel text display for exclusive use by an application.

### Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString,
wait, wait, wait
```

## Constructor Detail

### FrontPanelManager

protected **FrontPanelManager**()

Protected constructor. Cannot be used by an application.

## Method Detail

### getInstance

public static **FrontPanelManager** **getInstance**()

Gets the singleton instance of the front panel manager. The singleton MAY be implemented using application or implementation scope.

**Returns:**

The front panel manager.

**Throws:**

`java.lang.SecurityException` - if the calling application does not have `MonitorAppPermission("frontpanel")`.

### reserveTextDisplay

public boolean **reserveTextDisplay**(ResourceClient resourceClient)

Reserves the front panel text display for exclusive use by an application. The ResourceProxy of the TextDisplay SHALL be used for resource contention.

**Parameters:**

`resourceClient` - A DAVIC resource client for resource control.

**Returns:**

True if the implementation accepted the reservation request, otherwise returns false.

### reserveIndicator

public boolean **reserveIndicator**(ResourceClient resourceClient,  
java.lang.String indicator)

Reserves one of the indicators for exclusive use by an application. The ResourceProxy of the Indicator SHALL be used for resource contention.

**Parameters:**

`resourceClient` - A DAVIC resource client for resource control.

`indicator` - One of the indicator String names found in the table returned by the `getSupportedIndicators()` method.

**Returns:**

True if the implementation accepted the reservation request, otherwise returns false.

**Throws:**

`java.lang.IllegalArgumentException` - if the indicator argument is not contained in the table returned by the `getSupportedIndicators()` method.

### releaseTextDisplay

public void **releaseTextDisplay**()

Releases the front panel text display from a previous reservation. If the calling application is not the application that reserved the front panel, or if the front panel is not reserved when this method is called, this method does nothing.

**releaseIndicator**

```
public void releaseIndicator(java.lang.String indicator)
```

Releases a front panel indicator from a previous reservation. If the calling application is not the application that reserved the indicator, or if the indicator is not reserved when this method is called, this method does nothing.

**Parameters:**

`indicator` - One of the indicator String names found in the table returned by the `getSupportedIndicators()` method.

**Throws:**

`java.lang.IllegalArgumentException` - if the indicator argument is not contained in the table returned by the `getSupportedIndicators()` method.

**getTextDisplay**

```
public TextDisplay getTextDisplay()
```

Gets the front panel text display. A front panel must be reserved before an application can get it with this method.

**Returns:**

Front panel text display, or null if the application has not reserved it.

**getSupportedIndicators**

```
public java.lang.String[] getSupportedIndicators()
```

Gets the set of available indicators. The array returned SHALL contain the name of all the indicators supported with `getIndicatorDisplay(String[])`. The set of standardized indicators includes "power", "rbyypass", "message", and "record" and these MAY be returned.

**Returns:**

The set of supported indicators. MAY return indicators not included in the standardized set.

**getIndicatorDisplay**

```
public IndicatorDisplay getIndicatorDisplay(java.lang.String[] indicators)
```

Gets the individual indicators display. Indicators must be reserved before an application can get them with this method.

**Parameters:**

`indicators` - Set of indicator names.

**Returns:**

Set of individual indicators, or null if the application has not reserved one or more of the parameter indicators.

**Throws:**

`java.lang.IllegalArgumentException` - if any of the indicator arguments are not contained in the table returned by the `IndicatorDisplay.getIndicators()` method.

**isTextDisplaySupported**

```
public boolean isTextDisplaySupported()
```

Provides an indication of whether or not text display is supported. For backwards compatibility, the implementation must allow an application to reserve, get, and write to the text display even if the device does not contain a text display. If an application attempts to write to a nonexistent text display, the implementation shall not return an error.

**Returns:**

True if text display is supported, false if text display is not supported.

## org.ocap.hardware.frontpanel Interface Indicator

**All Superinterfaces:**  
ResourceProxy

```
public interface Indicator
extends ResourceProxy
```

This interface represents an indicator in the front panel display and allows its properties to be checked and set.

### Method Summary

BlinkSpec	<b>getBlinkSpec()</b> Gets the blink specification for the front panel Indicator.
BrightSpec	<b>getBrightSpec()</b> Gets the brightness specification for the front panel Indicator.
ColorSpec	<b>getColorSpec()</b> Gets the Color specification for the front panel Indicator.
void	<b>setBlinkSpec(BlinkSpec blinkSpec)</b> Sets the blink specification for the front panel Indicator.
void	<b>setBrightSpec(BrightSpec brightSpec)</b> Sets the Brightness specification for the front panel Indicator.
void	<b>setColorSpec(ColorSpec colorSpec)</b> Sets the Color specification for the front panel Indicator.

### Methods inherited from interface org.davic.resources.ResourceProxy

getClient

### Method Detail

#### getBrightSpec

BrightSpec **getBrightSpec()**

Gets the brightness specification for the front panel Indicator. Changing values within the object returned by this method does not take affect until one of the set methods in this interface is called and the object is passed to the implementation.

**Returns:**

LED front panel brightness specification.

#### setBrightSpec

void **setBrightSpec(BrightSpec brightSpec)**  
throws java.lang.IllegalStateException

Sets the Brightness specification for the front panel Indicator.

**Parameters:**

brightSpec - Brightness specification.

**Throws:**

`java.lang.IllegalArgumentException` - if null is passed in.

`java.lang.IllegalStateException` - if the Indicator resource was lost.

**getColorSpec**

`ColorSpec` **getColorSpec()**

Gets the Color specification for the front panel Indicator. Changing values within the object returned by this method does not take affect until one of the set methods in this interface is called and the object is passed to the implementation.

**Returns:**

LED front panel Color specification or MAY return null if changing the color is not supported.

**setColorSpec**

void **setColorSpec**(`ColorSpec` colorSpec)  
throws `java.lang.IllegalStateException`

Sets the Color specification for the front panel Indicator.

**Parameters:**

colorSpec - Color specification if color is desired.

**Throws:**

`java.lang.IllegalArgumentException` - if null is passed in.

`java.lang.IllegalStateException` - if the Indicator resource was lost.

**getBlinkSpec**

`BlinkSpec` **getBlinkSpec()**

Gets the blink specification for the front panel Indicator. Changing values within the object returned by this method does not take affect until one of the set display methods in this interface is called and the object is passed to the implementation.

**Returns:**

LED front panel blink specification or MAY return null if blinking is not supported.

**setBlinkSpec**

void **setBlinkSpec**(`BlinkSpec` blinkSpec)  
throws `java.lang.IllegalStateException`

Sets the blink specification for the front panel Indicator.

**Parameters:**

blinkSpec - Blink specification if blinking is desired. A value of null turns blinking off.

**Throws:**

`java.lang.IllegalStateException` - if the Indicator resource was lost.

## org.ocap.hardware.frontpanel Interface IndicatorDisplay

```
public interface IndicatorDisplay
```

This interface represents the set of individual indicators on a front panel (e.g. power, recording). An individual indicator is a single lamp or icon that can be turned on or off. It may have properties such as color and brightness that can be set.

<b>Field Summary</b>	
static java.lang.String	<b>MESSAGE</b> Message LED
static java.lang.String	<b>POWER</b> Power LED
static java.lang.String	<b>RECORD</b> Record LED
static java.lang.String	<b>REMOTE</b> Remote LED
static java.lang.String	<b>RFBYPASS</b> RF bypass LED

<b>Method Summary</b>	
java.util.Hashtable	<b>getIndicators()</b> Gets the set of reserved indicators.

## **Field Detail**

### **POWER**

```
static final java.lang.String POWER  
    Power LED
```

### **RFBYPASS**

```
static final java.lang.String RFBYPASS  
    RF bypass LED
```

### **MESSAGE**

```
static final java.lang.String MESSAGE  
    Message LED
```

### **RECORD**

```
static final java.lang.String RECORD
```

Record LED

## REMOTE

```
static final java.lang.String REMOTE
    Remote LED
```

## Method Detail

### getIndicators

```
java.util.Hashtable getIndicators()
```

Gets the set of reserved indicators. The HashTable returned SHALL contain the name of the indicator and a corresponding Indicator object. The set of standardized indicators includes "power", "rfbypass", "message", and "record". The Indicator object associated with each indicator can be used to change the color and brightness (i.e. turn on or off) if those properties are supported.

**Returns:**

The set of Indicators reserved using  
`FrontPanelManager.reserveIndicator(org.davic.resources.ResourceClient,  
java.lang.String)`

**org.ocap.hardware.frontpanel**  
**Interface ScrollSpec**

```
public interface ScrollSpec
```

This interface represents the scrolling specification of a front panel text display. In a single line display text will scroll from right to left. In a multi-line display text will scroll from bottom to top. In this mode either text lines must not exceed the length of the display or wrap must be turned on.

## Method Summary

int	<b>getHoldDuration()</b> Gets the percentage of time the scroll will hold at each character during one scroll iteration.
int	<b>getHorizontalIterations()</b> Gets the number of times per minute the characters are set to scroll across the screen from right to left.
int	<b>getMaxHorizontalIterations()</b> Gets the maximum number of times per minute characters can scroll right to left across the display with zero hold time set.
int	<b>getMaxVerticalIterations()</b> Gets the maximum number of times per minute characters can scroll bottom to top across the display with zero hold time set.
int	<b>getVerticalIterations()</b> Gets the number of times per minute the characters are set to scroll across the screen from bottom to top.
void	<b>setHoldDuration(int duration)</b> Sets the percentage of time to hold at each character before scrolling it to the next position during one scroll iteration.
void	<b>setHorizontalIterations(int iterations)</b> Sets the number of times per minute one character will scroll across the display from right to left.
void	<b>setVerticalIterations(int iterations)</b> Sets the number of times per minute one character will scroll across the display from bottom to top.

## Method Detail

### getMaxHorizontalIterations

```
int getMaxHorizontalIterations()
```

Gets the maximum number of times per minute characters can scroll right to left across the display with zero hold time set.

**Returns:**  
Number of horizontal scroll iterations per minute. Returns zero if horizontal scroll is not supported.

### getMaxVerticalIterations

```
int getMaxVerticalIterations()
```

Gets the maximum number of times per minute characters can scroll bottom to top across the display with zero hold time set.

**Returns:**

Number of vertical scroll iterations per minute. Returns -1 if the display only supports one row. Returns zero if vertical scroll is not supported.

### **getHorizontalIterations**

`int getHorizontalIterations()`

Gets the number of times per minute the characters are set to scroll across the screen from right to left.

**Returns:**

Number of horizontal scroll iterations per minute. A value of 0 indicates horizontal scrolling is turned off. A value of -1 indicates there is more than one row displayed and characters will scroll vertically.

### **getVerticalIterations**

`int getVerticalIterations()`

Gets the number of times per minute the characters are set to scroll across the screen from bottom to top.

**Returns:**

Number of vertical scroll iterations per minute. A value of 0 indicates vertical scrolling is turned off. A value of -1 indicates there is only one row of characters displayed and characters will scroll horizontally.

### **setHorizontalIterations**

`void setHorizontalIterations(int iterations)`

Sets the number of times per minute one character will scroll across the display from right to left.

**Parameters:**

`iterations` - Number of horizontal scroll iterations per minute.

**Throws:**

`java.lang.IllegalArgumentException` - if the iteration is negative or exceed the value returned by `getMaxHorizontalIterations`.

### **setVerticalIterations**

`void setVerticalIterations(int iterations)`

Sets the number of times per minute one character will scroll across the display from bottom to top.

**Parameters:**

`iterations` - Number of vertical scroll iterations per minute.

**Throws:**

`java.lang.IllegalArgumentException` - if the iteration is negative or exceed the value returned by `getMaxVerticalIterations`.

### **getHoldDuration**

`int getHoldDuration()`

Gets the percentage of time the scroll will hold at each character during one scroll iteration.

**Returns:**

Character hold duration.

### **setHoldDuration**

`void setHoldDuration(int duration)`

Sets the percentage of time to hold at each character before scrolling it to the next position during one scroll iteration.

**Parameters:**

`duration` - Character hold percentage duration. Setting this value causes a smooth scroll across all characters without a hold on any of them.

**Throws:**

`java.lang.IllegalArgumentException` - if `duration` is negative or if the duration percentage is greater than 100 divided by the number of characters to scroll across during horizontal scroll, or the number of rows to scroll across during vertical scroll.

## org.ocap.hardware.frontpanel Interface TextDisplay

**All Superinterfaces:**  
ResourceProxy

```
public interface TextDisplay
extends ResourceProxy
```

This interface represents one line of characters in a front panel display.

### Field Summary

static byte	<b>STRING_MODE</b> The line can be set using a string of displayable characters.
static byte	<b>TWELVE_HOUR_CLOCK</b> This line will display the network time using a standard 12 hour HH:MM format.
static byte	<b>TWENTYFOUR_HOUR_CLOCK</b> This line will display the network time using a military 24 hour HH:MM format.

### Method Summary

void	<b>eraseDisplay()</b> Removes characters from the text display.
BlinkSpec	<b>getBlinkSpec()</b> Gets the blink specification for the front panel text display.
BrightSpec	<b>getBrightSpec()</b> Gets the brightness specification for the front panel text display.
java.lang.String	<b>getCharacterSet()</b> Gets the set of characters supported by the display.
ColorSpec	<b>getColorSpec()</b> Gets the Color specification for the front panel text display.
int	<b>getMode()</b> Gets the text display mode.
int	<b>getNumberColumns()</b> Gets the number of columns (characters) per line in the text display.
int	<b>getNumberRows()</b> Gets the number of rows (i.e. lines) in the text display.
ScrollSpec	<b>getScrollSpec()</b> Gets the scroll specification for the front panel text display.
void	<b>setClockDisplay(byte mode, BlinkSpec blinkSpec, ScrollSpec scrollSpec, BrightSpec brightSpec, ColorSpec colorSpec)</b> Displays the current system time on the front panel text display.

## Method Summary

void	<b>setTextDisplay</b> (java.lang.String[] text, BlinkSpec blinkSpec, ScrollSpec scrollSpec, BrightSpec brightSpec, ColorSpec colorSpec) Displays text on the front panel display.
void	<b>setWrap</b> (boolean wrap) Sets wrapping on or off.

### Methods inherited from interface org.davic.resources.ResourceProxy

getClient

## Field Detail

### TWELVE\_HOUR\_CLOCK

static final byte **TWELVE\_HOUR\_CLOCK**

This line will display the network time using a standard 12 hour HH:MM format.

### TWENTYFOUR\_HOUR\_CLOCK

static final byte **TWENTYFOUR\_HOUR\_CLOCK**

This line will display the network time using a military 24 hour HH:MM format.

### STRING\_MODE

static final byte **STRING\_MODE**

The line can be set using a string of displayable characters.

## Method Detail

### getBrightSpec

BrightSpec **getBrightSpec**()

Gets the brightness specification for the front panel text display. Changing values within the object returned by this method does not take affect until one of the set methods in this interface is called and the object is passed to the implementation.

**Returns:**

LED front panel brightness specification.

### getColorSpec

ColorSpec **getColorSpec**()

Gets the Color specification for the front panel text display. Changing values within the object returned by this method does not take affect until one of the set methods in this interface is called and the object is passed to the implementation.

**Returns:**

LED front panel Color specification or MAY return null if changing the color is not supported.

### getBlinkSpec

BlinkSpec **getBlinkSpec**()

Gets the blink specification for the front panel text display. Changing values within the object returned by this method does not take affect until one of the set display methods in this interface is called and the object is passed to the implementation.

**Returns:**

LED front panel blink specification or MAY return null if blinking is not supported.

### **getScrollSpec**

ScrollSpec **getScrollSpec()**

Gets the scroll specification for the front panel text display. Changing values within the object returned by this method does not take affect until one of the set display methods in this interface is called and the object is passed to the implementation.

**Returns:**

LED front panel scroll specification.

### **getMode**

int **getMode()**

Gets the text display mode. See definitions of TWELVE\_HOUR\_CLOCK, TWENTYFOUR\_HOUR\_CLOCK, and STRING\_MODE for possible return values.

**Returns:**

Text display mode.

### **getNumberColumns**

int **getNumberColumns()**

Gets the number of columns (characters) per line in the text display. The text is considered fixed font by this method. Dynamic font sizes can be supported and the calculation for this method uses the largest character size for the given font.

**Returns:**

Number of columns.

### **getNumberRows**

int **getNumberRows()**

Gets the number of rows (i.e. lines) in the text display.

**Returns:**

Number of rows.

### **getCharacterSet**

java.lang.String **getCharacterSet()**

Gets the set of characters supported by the display. This API does not contain font support and this method is the only way to discover the character set supported by the front panel. In addition, certain types of displays do not support the entire alphabet or symbol set, e.g. seven segment LEDs.

**Returns:**

Supported character set.

### **setClockDisplay**

```
void setClockDisplay(byte mode,  
                    BlinkSpec blinkSpec,  
                    ScrollSpec scrollSpec,  
                    BrightSpec brightSpec,  
                    ColorSpec colorSpec)
```

throws `java.lang.IllegalStateException`

Displays the current system time on the front panel text display. The display is formatted to the mode parameter.

**Parameters:**

`mode` - One of the clock modes; `TWELVE_HOUR_CLOCK`, or `TWENTYFOUR_HOUR_CLOCK`.  
`blinkSpec` - Blink specification if blinking is desired. A value of null turns blinking off.  
`scrollSpec` - Scroll specification if scrolling is desired. A value of null turns scrolling off. If there is only one line of text scrolling will be from right to left. If there is more than one line of text scrolling will be from bottom to top. Passing in null turns scrolling off.  
`brightSpec` - Brightness specification if a change in brightness is desired. A value of null results in no change to current brightness.  
`colorSpec` - Color specification if a change in color is desired. A value of null results in no change to current color.

**Throws:**

`java.lang.IllegalArgumentException` - if the mode parameter is not one of `TWELVE_HOUR_CLOCK`, `TWENTYFOUR_HOUR_CLOCK`.  
`java.lang.IllegalStateException` - if the `TextDisplay` resource was lost.

### setTextDisplay

```
void setTextDisplay(java.lang.String[] text,
                   BlinkSpec blinkSpec,
                   ScrollSpec scrollSpec,
                   BrightSpec brightSpec,
                   ColorSpec colorSpec)
    throws java.lang.IllegalStateException
```

Displays text on the front panel display. If multiple fonts are possible the implementation SHALL determine which will be used. Sets the LED front panel to the text mode; see `STRING_MODE`. The text parameter will be used to display text characters in the display. Wrapping occurs if there is more than one line, wrapping is turned on, and the text over-fills at least one line.

**Parameters:**

`text` - String of characters to display. Each string in the array represents a line of text. `text[0]` represents the top line, `text[1]` represents the next line down, and so forth.  
`blinkSpec` - Blink specification if blinking is desired. Passing in null turns blinking off.  
`scrollSpec` - Scroll specification if scrolling is desired. If there is only one line of text scrolling will be from right to left. If there is more than one line of text scrolling will be from bottom to top. Passing in null turns scrolling off.  
`brightSpec` - Brightness specification if a change in brightness is desired. A value of null results in no change to current brightness.  
`colorSpec` - Color specification if a change in color is desired. A value of null results in no change to current color.

**Throws:**

`java.lang.IllegalArgumentException` - if the text array contains more than 1 line and one or more lines are longer than the display and wrap is turned off.  
`java.lang.IllegalStateException` - if the `TextDisplay` resource was lost.

### setWrap

```
void setWrap(boolean wrap)
```

Sets wrapping on or off.

**Parameters:**

`wrap` - If wrap is true wrapping is turned on, otherwise wrapping is turned off.

**eraseDisplay**

```
void eraseDisplay()
```

Removes characters from the text display.

## Appendix I      Revision History

The following ECNs were incorporated into OC-SP-OCAP-FPEXT-I02-071220:

<b>ECN</b>	<b>Date Accepted</b>	<b>Title of EC</b>
OCAP-FPEXT-N-05.0843-1	12/19/05	System property identifying Front Panel extension
OCAP-FPEXT-N-05.0848-1	1/3/06	Front Panel Registry of Constants update
OCAP-FPEXT-N-05.0851-2	1/31/06	FrontPanel ECR
OCAP-FPEXT-N-06.0864-1	2/7/06	Correction of System property identifying Front Panel extension
OCAP-FPEXT-N-06.0946-1	11/16/06	Correct constant of bright level
OCAP-FPEXT-N-06.0956-1	12/5/06	Remove a security exception from a ColorSpec.setColor() method
OCAP-FPEXT-N-07.1058-1	7/16/07	Correct ocap:monitorapplication reference
OCAP-FPEXT-N-07.1097-1	10/16/07	Update BrightSpec to use consistent types

The following ECN was incorporated into OC-SP-OCAP-FPEXT-I03-090612:

<b>ECN</b>	<b>Date Accepted</b>	<b>Title of EC</b>
OCAP-FPEXT-N-09.1414-1	6/12/09	Reference cleanup

The following ECN was incorporated into OC-SP-OCAP-FPEXT-I04-100603:

<b>ECN</b>	<b>Date Accepted</b>	<b>Title of EC</b>
OCAP-FPEXT-N-09.1470-3	6/3/10	Optional Front Panel Text Display