

OpenCable™ Application Platform Specification

Front Panel Extension

OC-SP-OCAP-FPEXT-I01-050624

**ISSUED
SPECIFICATION**

Notice

This document 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 the 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, or fitness for a particular purpose of this document, or any document referenced herein.

If you, or the company or organization that you represent, has an existing agreement with CableLabs concerning the use of this document, or subsequently enters into another agreement with CableLabs concerning the use of this document (e.g., the OCAP Implementers Agreement), your rights and obligations with respect to this document, and the rights and obligations of the company or organization that you represent, SHALL be as set forth therein.

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

Document Status Sheet

Document Control Number:	OC-SP-OCAP-FPEXT-I01-050624			
Document Series:	OpenCable™ Application Platform Specification			
Reference:	OCAP 1.0 Profile			
Revision History:	D01-050325, Draft Specification D02-050610, Draft Specification I01-050624, Issued Specification			
Date:	June 24, 2005			
Status Code:	Work in Process	Draft	Issued	Closed
Distribution Restrictions:	Author Only	CL/Member	CL/Member/Vendor	Public

Key to Document Status Codes

- Work in Process** 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 cable industry 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:

DOCSIS®, eDOCSIS™, PacketCable™, CableHome®, CableOffice™, OpenCable™, CableCARD™, and CableLabs® are trademarks of Cable Television Laboratories, Inc.

Table of Contents

1 SCOPE	1
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 Specification Language	2
2 REFERENCES.....	3
2.1 DVB-GEM AND DVB-MHP SPECIFICATION CORRESPONDENCE	3
2.2 NORMATIVE REFERENCES	3
2.3 INFORMATIVE REFERENCES.....	3
3 TERMS AND DEFINITIONS	5
4 ACRONYMS.....	7
5 FRONT PANEL	9
5.1 INTRODUCTION	9
5.1.1 OpenCable Host Front Panel Example (Informative).....	9
5.2 MODES OF DISPLAY	9
5.3 INDICATORS	9
5.4 SECURITY	10
ANNEX A FRONT PANEL APIS ORG.OCAP.HARDWARE.FRONTPANEL	11
org.ocap.hardware.frontpanel	11
BlinkSpec	12
getIterations()	12
getMaxCycleRate().....	12
getOnDuration()	12
setIterations(int).....	13
setOnDuration(int)	13
FrontPanelManager	14
FrontPanelManager()	14
getIndicatorDisplay(String[])	15
getInstance().....	15
getTextDisplay()	15
releaseIndicator(String)	15
releaseTextDisplay().....	15
reserveIndicator(ResourceClient, String)	16
reserveTextDisplay(ResourceClient).....	16
Indicator	17
BLUE	18
COLOR_NOT_SUPPORTED.....	18

GREEN	18
OFF	18
ORANGE	18
RED	18
YELLOW	18
getBrightness().....	18
getBrightnessLevels()	18
getColor()	19
getSupportedColors().....	19
setBrightness(byte)	19
setColor(byte)	19
IndicatorDisplay	20
MESSAGE.....	20
POWER.....	20
RECORD.....	20
REMOTE	21
RFBYPASS	21
getIndicators()	21
IndicatorProperties	22
BLUE	22
COLOR_NOT_SUPPORTED	23
GREEN	23
OFF.....	23
ORANGE	23
RED	23
YELLOW	23
getBrightness()	23
getBrightnessLevels()	23
getColor().....	23
getSupportedColors()	24
setBrightness(byte).....	24
setColor(byte)	24
ScrollSpec	25
getHoldDuration().....	25
getHorizontalIterations().....	26
getMaxHorizontalIterations()	26
getMaxVerticalIterations()	26
getVerticalIterations().....	26
setHoldDuration(int)	26
setHorizontalIterations(int)	26
setVerticalIterations(int)	27
TextDisplay	28
STRING_MODE	29
TWELVE_HOUR_CLOCK	29
TWENTYFOUR_HOUR_CLOCK.....	29
eraseDisplay().....	29

getBlinkSpec()	29
getCharacterSet()	29
getMode()	29
getNumberColumns()	30
getNumberRows()	30
getScrollSpec()	30
setClockDisplay(byte, BlinkSpec, ScrollSpec).....	30
setTextDisplay(String[], BlinkSpec, ScrollSpec)	30
setWrap(boolean)	31

This page intentionally left blank.

List of Figures

FIGURE 5-1: 4 DIGIT, 7-SEGMENT LED DISPLAY	9
--------------------------------------------------	---

This page intentionally left blank.

List of Tables

TABLE 5-1 PERMISSIONS10

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 1.0 (OCAP 1.0). The OCAP 1.0 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 1.0 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 which use FP on all OpenCable-compliant host devices connected to their networks, which contains 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 **MUST** 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 **MUST** be agnostic as to the type of front panel hardware (e.g., 7 segment LED, LCD).
- The front panel API **MUST** 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 p`
- **Boldfaced** text is used as emphasis.
- *Italicized* text is used to indicate section titles when referenced within the text.

1.4 Front Panel Specific Requirements

1.4.1 Specification Language

The following words are used throughout this document to define the significance of particular requirements:

MUST/SHALL	This word, or the adjective REQUIRED, means that the item is an absolute requirement of this specification.
MUST NOT/SHALL NOT	This phrase means that the item is an absolute prohibition of this specification.
SHOULD	This word, or the adjective RECOMMENDED, means that valid reasons may exist 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, or the adjective OPTIONAL, 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.

Note: Other text is descriptive or explanatory.

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.

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.

A non-specific reference to an ETS SHALL also be taken to refer to later versions published as an EN with the same number.

Some known errata in these references are identified in Annex A of DVB-MHP 1.0.3 [3]. These errata take precedence over the published reference.

OCAP 1.0 Index	References	Edition	Description
[1]	OCAP 1.0	OC-SP-OCAP1.0-115-050415 April 15, 2005	OpenCable Application Platform http://www.opencable.com/specifications/

2.3 Informative References

OCAP 1.0 index	Reference	Edition	Description
[2]	DVB-GEM 1.0.2	May 2005	DVB Globally Executable MHP 1.0.2 http://222/mhp.org/mhp_technology/gem/ Note: At the time of the publication of this document, the above reference is only available as a DVB Bluebook, A089. It will become available from ETSI at the completion of the approval process.
[3]	DVB-MHP 1.0.3	ETSI TS 101 812 v 1.3.1 June 2003	DVB Multimedia Home Platform 1.0.3 http://www.etsi.org/services_products/freestandard/home.htm

This page intentionally left blank.

3 Terms and Definitions

The following definitions are used in this specification:

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.

This page intentionally left blank.

4 Acronyms

The following acronyms are used in this specification:

DTD	Document Type Definition
FP	Front Panel
PRF	Permission Request File

This page intentionally left blank.

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, *Front Panel APIs org.ocap.hardware.frontpanel*. 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)

The Host is designed with a front panel display that incorporates at least a POWER LED and MESSAGE LED, and at least 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:

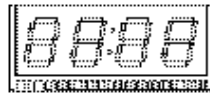


Figure 5-1: 4 Digit, 7-Segment LED Display

The Host is designed with a front panel display and includes RF Bypass functionality and a RF BYPASS LED.

This LED is active when the RF Bypass is active. If the Host is designed with a front panel display, it will incorporate the OCAP extensions defined in this document.

5.2 Modes of 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 a minimum set of indicator lamps, typically LEDs, including power, RF bypass, message, and record. An application SHALL be able to turn these lamps on and off. The ability to change color and brightness support is optional.

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 DTD of the PRF for such Host devices SHALL contain a “frontpanel” entry in the `OCAP:MonitorAppPermission` element.

Annex A Front Panel APIs

org.ocap.hardware.frontpanel

This annex describes the Front Panel APIs.

Package

org.ocap.hardware.frontpanel

Class Summary	
Interfaces	
BlinkSpec	This interface represents the front panel text display blinking specification.
Indicator	This interface represents an indicator in the front panel display and allows its properties to be checked and set.
IndicatorDisplay	This interface represents the set of individual indicators on a front panel (e.g.,
IndicatorProperties	This interface represents properties of an indicator in the front panel display, including color and brightness.
ScrollSpec	This interface represents the scrolling specification of a front panel text display.
TextDisplay	This interface represents one line of characters in a front panel display.
Classes	
FrontPanelManager	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

BlinkSpec

Declaration

```
public interface BlinkSpec
```

Description

This interface represents the front panel text display blinking specification. All characters blink at the same time.

Member Summary

Methods

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

Methods

getIterations()

```
public int getIterations()
```

Gets the number of times per minute the text display will blink.

Returns: Number of blink iterations per minute.

getMaxCycleRate()

```
public 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.

getOnDuration()

```
public int getOnDuration()
```

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

Returns: Text display blink on percentage duration.

setIterations(int)

```
public 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.

setOnDuration(int)

```
public 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 FrontPanelManager

Declaration

```
public class FrontPanelManager
    java.lang.Object
    |
    +--org.ocap.hardware.frontpanel.FrontPanelManager
```

Description

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.

Member Summary	
Constructors	
protected	FrontPanelManager() Protected constructor.
Methods	
IndicatorDisplay	getIndicatorDisplay(java.lang.String indicators) Gets the individual indicators display.
static	getInstance() Gets the singleton instance of the front panel manager.
FrontPanelManager	
TextDisplay	getTextDisplay() Gets the front panel text display.
void	releaseIndicator(java.lang.String indicator) Releases a front panel indicator from a previous reservation.
void	releaseTextDisplay() Releases the front panel text display from a previous reservation.
boolean	reserveIndicator(org.davic.resources.ResourceClient resourceClient, java.lang.String indicator) Reserves one of the indicators for exclusive use by an application.
boolean	reserveTextDisplay(org.davic.resources.ResourceClient resourceClient) Reserves the front panel text display for exclusive use by an application.

Constructors

FrontPanelManager()

```
protected FrontPanelManager()
```

Protected constructor. Cannot be used by an application.

Methods

getIndicatorDisplay(String[])

```
public org.ocap.hardware.frontpanel.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.

getInstance()

```
public static org.ocap.hardware.frontpanel.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")`.

getTextDisplay()

```
public org.ocap.hardware.frontpanel.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.

releaseIndicator(String)

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

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.

Throws:

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

reserveIndicator(ResourceClient, String)

```
public boolean reserveIndicator(org.davic.resources.ResourceClient resourceClient,  
    java.lang.String indicator)  
    throws IllegalArgumentException
```

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 `IndicatorDisplay.getIndicators()` method.

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

Throws:

`java.lang.IllegalArgumentException` - if indicator does not equal one of the indicator names.

reserveTextDisplay(ResourceClient)

```
public boolean reserveTextDisplay(org.davic.resources.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.

org.ocap.hardware.frontpanel Indicator

Declaration

```
public interface Indicator extends org.davic.resources.ResourceProxy
```

All Superinterfaces: org.davic.resources.ResourceProxy

Description

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

Member Summary	
Fields	
static byte	BLUE Indicator color blue.
static byte	COLOR_NOT_SUPPORTED Indicator color not supported.
static byte	GREEN Indicator color green.
static byte	OFF Minimum brightness setting.
static byte	ORANGE Indicator color orange.
static byte	RED Indicator color red.
static byte	YELLOW Indicator color yellow.
Methods	
int	getBrightness() Gets the brightness of the Indicator.
int	getBrightnessLevels() Gets the number of brightness levels supported.
byte	getColor() Gets the current color of the indicator.
byte	getSupportedColors() Gets the supported colors.
void	setBrightness(byte brightness) Sets the brightness of the indicator.
void	setColor(byte color) Sets the color of this indicator.

Fields

BLUE

```
public static final byte BLUE
```

Indicator color blue.

COLOR_NOT_SUPPORTED

```
public static final byte COLOR_NOT_SUPPORTED
```

Indicator color not supported.

GREEN

```
public static final byte GREEN
```

Indicator color green.

OFF

```
public static final byte OFF
```

Minimum brightness setting.

ORANGE

```
public static final byte ORANGE
```

Indicator color orange.

RED

```
public static final byte RED
```

Indicator color red.

YELLOW

```
public static final byte YELLOW
```

Indicator color yellow.

Methods

getBrightness()

```
public int getBrightness()
```

Gets the brightness of the Indicator. Possible return values include OFF and any brightness levels above that up to the value returned by `getBrightnessLevels() + 1`.

Returns: LED brightness.

getBrightnessLevels()

```
public int getBrightnessLevels()
```

Gets the number of brightness levels supported. The minimum support SHALL be OFF + 1. This provides an on/off capability.

Returns: Indicator brightness levels.

getColor()

```
public byte getColor()
```

Gets the current color of the indicator. See definitions of COLOR_NOT_SUPPORTED, BLUE, GREEN, RED, YELLOW, and ORANGE for possible values.

Returns: Indicator color.

getSupportedColors()

```
public byte getSupportedColors()
```

Gets the supported colors. The value returned SHALL contain values for the possible color set OR'ed together. See definitions of COLOR_NOT_SUPPORTED, BLUE, GREEN, RED, YELLOW, and ORANGE for possible values.

Returns: Supported color set. Returns COLOR_NOT_SUPPORTED if colors are not supported.

setBrightness(byte)

```
public void setBrightness(byte brightness)
           throws 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 one of OFF to the value returned by getBrightnessLevels() + 1.

setColor(byte)

```
public void setColor(byte color)
           throws 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.

java.lang.SecurityException - if the caller does not have the indicator resource reserved.

org.ocap.hardware.frontpanel

IndicatorDisplay

Declaration

```
public interface IndicatorDisplay
```

Description

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.

Member Summary

Fields

```
static java.lang.String MESSAGE
                        Message LED
static java.lang.String POWER
                        Power LED
static java.lang.String RECORD
                        Record LED
static java.lang.String REMOTE
                        Remote LED
static java.lang.String RFBYPASS
                        RF bypass LED
```

Methods

```
java.util.Hashtable getIndicators()
                    Gets the set of available indicators.
```

Fields

MESSAGE

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

POWER

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

RECORD

```
public static final java.lang.String RECORD
Record LED
```

REMOTE

```
public static final java.lang.String REMOTE
```

Remote LED

RFBYPASS

```
public static final java.lang.String RFBYPASS
```

RF bypass LED

Methods

getIndicators()

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

Gets the set of available indicators. The `HashTable` returned SHALL contain the name of the indicator and a corresponding `IndicatorProperties` object. The set of standardized indicators includes “power”, “rfbypass”, “message”, and “record”. The `IndicatorProperties` 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 supported indicators. MAY return indicators not included in the standardized set.

org.ocap.hardware.frontpanel

IndicatorProperties

Declaration

```
public interface IndicatorProperties
```

Description

This interface represents properties of an indicator in the front panel display, including color and brightness.

Member Summary

Fields

```
static byte BLUE
    Indicator color blue.
static byte COLOR_NOT_SUPPORTED
    Indicator color not supported.
static byte GREEN
    Indicator color green.
static byte OFF
    Minimum brightness setting.
static byte ORANGE
    Indicator color orange.
static byte RED
    Indicator color red.
static byte YELLOW
    Indicator color yellow.
```

Methods

```
int getBrightness()
    Gets the brightness of the Indicator.
int getBrightnessLevels()
    Gets the number of brightness levels supported.
byte getColor()
    Gets the current color of the indicator.
byte getSupportedColors()
    Gets the supported colors.
void setBrightness(byte brightness)
    Sets the brightness of the indicator.
void setColor(byte color)
    Sets the color of all segments in all front panel LEDs.
```

Fields

BLUE

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

COLOR_NOT_SUPPORTED

```
public static final byte COLOR_NOT_SUPPORTED
```

Indicator color not supported.

GREEN

```
public static final byte GREEN
```

Indicator color green.

OFF

```
public static final byte OFF
```

Minimum brightness setting.

ORANGE

```
public static final byte ORANGE
```

Indicator color orange.

RED

```
public static final byte RED
```

Indicator color red.

YELLOW

```
public static final byte YELLOW
```

Indicator color yellow.

Methods

getBrightness()

```
public int getBrightness()
```

Gets the brightness of the Indicator. Possible return values include OFF and any brightness levels above that up to the value returned by `getBrightnessLevels() + 1`.

Returns: LED brightness.

getBrightnessLevels()

```
public int getBrightnessLevels()
```

Gets the number of brightness levels supported. The minimum support SHALL be OFF + 1. This provides an on/off capability.

Returns: Indicator brightness levels.

getColor()

```
public byte getColor()
```

Gets the current color of the indicator. See definitions of COLOR_NOT_SUPPORTED, BLUE, GREEN, RED, YELLOW, and ORANGE for possible values.

Returns: Indicator color.

getSupportedColors()

```
public byte getSupportedColors()
```

Gets the supported colors. The value returned SHALL contain values for the possible color set OR'ed together. See definitions of COLOR_NOT_SUPPORTED, BLUE, GREEN, RED, YELLOW, and ORANGE for possible values.

Returns: Supported color set.

setBrightness(byte)

```
public void setBrightness(byte brightness)
```

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 one of OFF to the value returned by getBrightnessLevels() + 1.

setColor(byte)

```
public void setColor(byte color)
```

Sets the color of all segments in all front panel LEDs.

Parameters:

color - Indicator color.

Throws:

java.lang.IllegalArgumentException - if the value of the color parameter is not in the supported color set.

java.lang.UnsupportedOperationException - if color is not supported.

org.ocap.hardware.frontpanel

ScrollSpec

Declaration

```
public interface ScrollSpec
```

Description

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.

Member Summary

Methods

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

Methods

getHoldDuration()

```
public int getHoldDuration()
```

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

Returns: Character hold duration.

getHorizontalIterations()

```
public 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.

getMaxHorizontalIterations()

```
public 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()

```
public 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.

getVerticalIterations()

```
public 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.

setHoldDuration(int)

```
public 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.

setHorizontalIterations(int)

```
public 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(int)

```
public 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`.

org.ocap.hardware.frontpanel TextDisplay

Declaration

```
public interface TextDisplay extends org.davic.resources.ResourceProxy
```

All Superinterfaces: org.davic.resources.ResourceProxy

Description

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

Member Summary	
Fields	
static byte	STRING_MODE The line can be set using a string of apparent characters.
static byte	TWELVE_HOUR_CLOCK This line will display the network time using a standard 12 hour HH:MM format.
static byte	TWENTYFOUR_HOUR_CLOCK This line will display the network time using a military 24 hour HH:MM format.
Methods	
void	eraseDisplay() Removes characters from the text display.
BlinkSpec	getBlinkSpec() Gets the blink specification for the front panel text display.
java.lang.String	getCharacterSet() Gets the set of characters supported by the display.
int	getMode() Gets the text display mode.
int	getNumberColumns() Gets the number of columns (characters) per line in the text display.
int	getNumberRows() Gets the number of rows in the text display.
ScrollSpec	getScrollSpec() Gets the scroll specification for the front panel text display.
void	setClockDisplay(byte mode, BlinkSpec blinkSpec, ScrollSpec scrollSpec) Displays the current system time on the front panel text display.
void	setTextDisplay(java.lang.String text, BlinkSpec blinkSpec, ScrollSpec scrollSpec) Displays text on the front panel display.
void	setWrap(boolean wrap) Sets wrapping on or off.

Fields

STRING_MODE

```
public static final byte STRING_MODE
```

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

TWELVE_HOUR_CLOCK

```
public static final byte TWELVE_HOUR_CLOCK
```

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

TWENTYFOUR_HOUR_CLOCK

```
public static final byte TWENTYFOUR_HOUR_CLOCK
```

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

Methods

eraseDisplay()

```
public void eraseDisplay()
```

Removes characters from the text display.

getBlinkSpec()

```
public org.ocap.hardware.frontpanel.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 scroll specification.

getCharacterSet()

```
public 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.

getMode()

```
public 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()

```
public 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()

```
public int getNumberRows()
```

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

Returns: Number of rows.

getScrollSpec()

```
public org.ocap.hardware.frontpanel.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.

setClockDisplay(byte, BlinkSpec, ScrollSpec)

```
public void setClockDisplay(byte mode, org.ocap.hardware.frontpanel.BlinkSpec  
    blinkSpec, org.ocap.hardware.frontpanel.ScrollSpec scrollSpec)
```

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.

Throws:

`java.lang.IllegalArgumentException` - if the mode parameter is not one of `TWELVE_HOUR_CLOCK`, `TWENTYFOUR_HOUR_CLOCK`.

setTextDisplay(String[], BlinkSpec, ScrollSpec)

```
public void setTextDisplay(java.lang.String[] text,  
    org.ocap.hardware.frontpanel.BlinkSpec blinkSpec,  
    org.ocap.hardware.frontpanel.ScrollSpec scrollSpec)
```

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.

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.

setWrap(boolean)

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

Sets wrapping on or off.

Parameters:

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

