

Cross Platform Extras Style Metadata

CONTENTS

| | | |
|-------|---|----|
| 1 | Introduction..... | 1 |
| 1.1 | Overview..... | 1 |
| 1.1.1 | Technical Approach..... | 1 |
| 1.1.2 | Extras Architecture..... | 1 |
| 1.2 | Document Organization..... | 1 |
| 1.3 | Document Notation and Conventions..... | 2 |
| 1.3.1 | XML Conventions..... | 2 |
| 1.3.2 | General Notes..... | 3 |
| 1.4 | Normative References..... | 4 |
| 1.5 | Informative References..... | 4 |
| 1.6 | General Types..... | 4 |
| 1.6.1 | ColorHex-type..... | 4 |
| 1.6.2 | ImageSubArea-type..... | 5 |
| 2 | Applying Styles to Experiences..... | 6 |
| 2.1 | Elements of Node Style..... | 6 |
| 2.2 | Node Style and Experience Hierarchy..... | 6 |
| 2.3 | Adaptive and Responsive Design..... | 7 |
| 2.4 | Putting it all together..... | 8 |
| 3 | CPE Style Top Level Elements..... | 10 |
| 3.1 | Mapping Experiences to Styles: ExperienceStyleMap-type..... | 11 |
| 3.1.1 | NodeStyleRef-type..... | 11 |
| 3.1.2 | CompatibilityDevice-type..... | 12 |
| 3.2 | Node Style (Experience Appearance)..... | 13 |
| 3.2.1 | NodeStyle-type..... | 13 |
| 4 | Themes..... | 15 |
| 4.1 | Theme-type..... | 15 |
| 4.2 | Color Palette..... | 15 |
| 4.2.1 | Neutral Palette..... | 15 |
| 4.2.2 | Colored Palettes..... | 16 |
| 4.2.3 | Base, Highlight and Defocus..... | 17 |
| 4.2.4 | ColorPalette-type..... | 18 |
| 4.2.5 | ColorPaletteInstance-type..... | 19 |
| 4.3 | Button Image Set..... | 19 |
| 4.3.1 | ButtonSet-type..... | 20 |
| 4.3.2 | Button-type..... | 20 |
| 4.3.3 | ButtonImages-type..... | 21 |
| 4.4 | Fonts..... | 21 |
| 4.4.1 | Text Rendering Flexibility..... | 21 |
| 4.4.2 | Text Rendering..... | 22 |
| 4.4.3 | Fonts-type..... | 22 |
| 5 | Selectable Objects: Buttons and Text..... | 24 |
| 5.1 | Built-in Buttons..... | 24 |

| | | |
|-------|---|----|
| 5.2 | Built-in Button Shapes | 25 |
| 5.3 | Image Buttons | 26 |
| 5.4 | Built-in Fonts..... | 26 |
| 5.5 | Button and Label backgrounds | 26 |
| 5.6 | Compositing..... | 26 |
| 6 | Background..... | 28 |
| 6.1 | Combining Elements into a Background..... | 28 |
| 6.2 | Background Assets..... | 28 |
| 6.2.1 | Background-type | 29 |
| 6.2.2 | Background Color..... | 29 |
| 6.3 | Background Image | 30 |
| 6.3.1 | BackgroundImage-type | 31 |
| 6.4 | Background Video | 32 |
| 6.4.1 | BackgroundVideo-type | 32 |
| 6.5 | Background Audio | 33 |
| 6.5.1 | BackgroundAudio-type | 33 |
| 6.6 | Adaptation | 33 |
| 6.6.1 | Adaptation-type | 36 |
| 6.7 | Overlay Areas..... | 37 |
| 6.7.1 | BackgroundOverlayArea-type | 38 |

REVISION HISTORY

| Version | Date | Description |
|---------|-----------------|--|
| V1.0 | August 15, 2016 | Initial version. Note: Version 1.0 is identical to the 8/15/16 draft. Publication as v1.0 occurred 1/3/17. |

1 INTRODUCTION

This document defines a method to control layout and appearance of a collection of content, such as a movie with extras. We defined ‘content’ broadly to include audiovisual content, galleries, applications and other media. We define ‘extras’ to mean anything offered in addition to the main title; sometimes called bonus material or value added material (VAM).

This document builds on Media Manifest Metadata by adding a simple menu system.

This specification is designed as a resource. Those using this specification may extend the definition with additional data element specific for their needs. They may replace elements with others perhaps more suitable to their needs; however, for interoperability all are highly encouraged to use the data elements exactly as defined.

Common Extras Metadata is part of the Common Metadata family of specifications.

1.1 Overview

1.1.1 Technical Approach

This document builds on Common Extras Manifest Metadata by describing how Manifest Experience elements are to be displayed.

1.1.2 Extras Architecture

The Extras Menu architecture has the following data objects

- [TBS]

From these components an Extras Menu can be created.

1.2 Document Organization

This document is organized as follows:

1. Introduction—Provides background, scope and conventions
2. Applying Styles to Experiences – Description of approach taken by this document
3. CPE Style Top Level Elements – Top level element definition and description
4. Themes – Description of style themes
5. Selectable Objects – Describes how buttons and text are presented for selection
6. Background – Screen background definition and background (what goes behind menus).

1.3 Document Notation and Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119]. That is:

- “MUST”, “REQUIRED” or “SHALL”, mean that the definition is an absolute requirement of the specification.
- “MUST NOT” or “SHALL NOT” means that the definition is an absolute prohibition of the specification.
- “SHOULD” or “RECOMMENDED” mean that there may be valid reasons to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- “SHOULD NOT” or “NOT RECOMMENDED” mean that there may be valid reasons when the particular behavior is acceptable, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- “MAY” or “OPTIONAL” mean the item is truly optional, however a preferred implementation may be specified for OPTIONAL features to improve interoperability.

Terms defined to have a specific meaning within this specification will be capitalized, e.g. “Track”, and should be interpreted with their general meaning if not capitalized.

Normative key words are written in all caps, e.g. “SHALL”

1.3.1 XML Conventions

XML is used extensively in this document to describe data. It does not necessarily imply that actual data exchanged will be in XML. For example, JSON may be used equivalently.

This document uses tables to define XML structure. These tables may combine multiple elements and attributes in a single table. Although this does not align with schema structure, it is much more readable and hence easier to review and to implement.

Although the tables are less exact than XSD, the tables should not conflict with the schema. Such contradictions should be noted as errors and corrected.

1.3.1.1 Naming Conventions

This section describes naming conventions for Common Metadata XML attributes, element and other named entities. The conventions are as follows:

- Names use initial caps, as in InitialCaps.

- Elements begin with a capital letter, as in `InitialCapitalElement`.
- Attributes begin with a lowercase letter, as in `initialLowercaseAttribute`.
- XML structures are formatted as Courier New, such as `md:rightstoken`
- Names of both simple and complex types are followed with “-type”

1.3.1.2 Structure of Element Table

Each section begins with an information introduction. For example, “The Bin Element describes the unique case information assigned to the notice.”

This is followed by a table with the following structure.

The headings are

- Element—the name of the element.
- Attribute—the name of the attribute
- Definition—a descriptive definition. The definition may define conditions of usage or other constraints.
- Value—the format of the attribute or element. Value may be an XML type (e.g., “string”) or a reference to another element description (e.g., “See Bar Element”). Annotations for limits or enumerations may be included (e.g., “int [0..100]” to indicate an XML `xs:int` type with an accepted range from 1 to 100 inclusively)
- Card—cardinality of the element. If blank, then it is 1. Other typical values are 0..1 (optional), 1..n and 0..n.

The first row of the table after the header is the element being defined. This is immediately followed by attributes of this element, if any. Subsequent rows are child elements and their attributes. All child elements (i.e., those that are direct descendents) are included in the table. Simple child elements may be fully defined here (e.g., “Title”, “”, “Title of work”, “`xs:string`”), or described fully elsewhere (“POC”, “”, “Person to contact in case there is a problem”, “`md:ContactInfo-type`”). In this example, if POC was to be defined by a complex type defined as `md:ContactInfo-type`. Attributes immediately follow the containing element.

Accompanying the table is as much normative explanation as appropriate to fully define the element, and potentially examples for clarity. Examples and other informative descriptive text may follow. XML examples are included toward the end of the document and the referenced web sites.

1.3.2 **General Notes**

All required elements and attributes must be included.

When enumerations are provided in the form ‘enumeration’, the quotation marks (‘’) should not be included.

The term “Device” refers to an entity playing the interactive material specified here. It may be a standalone physical device, such as a Blu-ray player, or it might be an application running on a general purpose computer, a table, phone or as part of another device. The term ‘User’ refers to the person using the Device.

1.4 Normative References

| | |
|----------------|--|
| [CM] | Common Metadata, TR-META-CM, www.movielabs.com/md/md |
| [Manifest] | MovieLabs Common Media Manifest Metadata v1.5, TR-META-MMM, www.movielabs.com/md/manifest |
| [CPE-Manifest] | Cross Platform Extras: CPE-Manifest, TR-CPE-M1, www.movielabs.com/cpe/manifest |
| [CPE-HTML] | Cross-Platform Extras API, TR-CPE-API, www.movielabs.com/md/cpe |
| [Avail] | EMA Content Availability Data (Avails), TR-META-AVAIL, www.movielabs.com/md/avails |
| [MEC] | Media Entertainment Core, TR-META-MEC, www.movielabs.com/md/mec |
| [CSS3FONT] | CSS Fonts Module Level 3, https://www.w3.org/TR/css-fonts-3/ |
| [CSS3COLOR] | CSS Color Module Level 3, https://www.w3.org/TR/css3-color/ |

1.5 Informative References

1.6 General Types

1.6.1 ColorHex-type

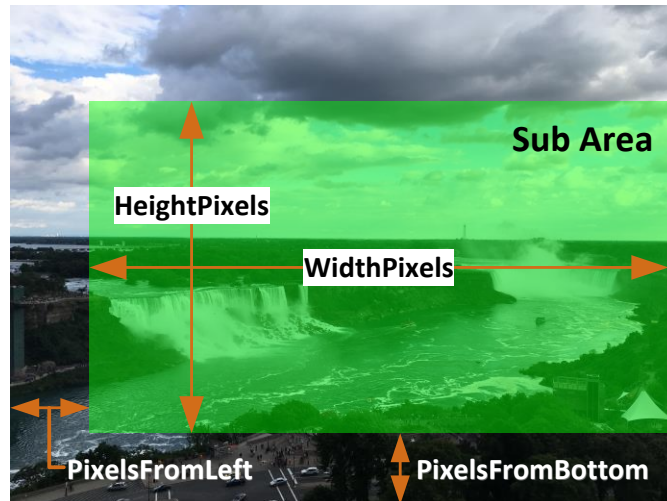
ColorHex-type is a string with the following pattern: `#[0-9a-fA-F]{6}`

ColorHex represents a color using hex representation of RGB color, as described in [CSS3COLOR].

1.6.2 ImageSubArea-type

Defines a subset of an image by area.

Parameters are shown in the following illustration:



| Element | Attribute | Definition | Value | Card. |
|--------------------------|-----------|---|-----------------------|-------|
| ImageSubArea-type | | | | |
| WidthPixels | | Width of subarea in pixels | xs:positiveInteger | 0..1 |
| HeighPixels | | Height of subarea in pixels | xs:positiveInteger | 0..1 |
| PixelsfromLeft | | Offset of the subarea from the left of the image in pixels. | xs:nonNegativeInteger | 0..1 |
| PlxelsFromBottom | | Offset of the subarea from the bottom of the image in pixels. | xs:nonNegativeInteger | |

2 APPLYING STYLES TO EXPERIENCES

The Appearance model provides a mechanism for specifying the visual and stylistic characteristics of Experience elements. This includes, but is not limited to, the use of colors, backgrounds, animation, and the customization of buttons and menu entries. Collectively, these characteristics are referred to as a Node Style. Section 2.1 describes the scope and contents of a Node Style and how Node Styles may be based on Themes. Section 2.2 explains how Node Styles are associated with specific Experiences. Finally, in Section 2.3, the ability of the Appearance model to support adaptive user interfaces that are sensitive to device orientation, category, or screen resolution is described.

2.1 Elements of Node Style

A Node Style is a set of appearance attributes that collectively define how Experiences appear. Certain Experience elements correspond with specific screens in the user interface. For example, there might be a top-level menu screen, “Extras”, “Featured Extras”, “Cast and Crew”, and so forth. The Appearance model is the mechanism for indicating the visual and stylistic attributes applicable to those screens, particularly background imagery and audio. The Node Style allows these elements to be specified and tied to the appropriate screen.

A Node Style may be applied to multiple Experiences which means that various places in a user interface will look similar. A key mechanism for specifying this type of common appearance is the use of Theme elements.

Generally, a Theme is applied across the application. The Theme will have a Color Palette and possibly a set of button images. Use of Theme colors and buttons ensures a unique and consistent use across the user experience, setting the right tone for the specific title.

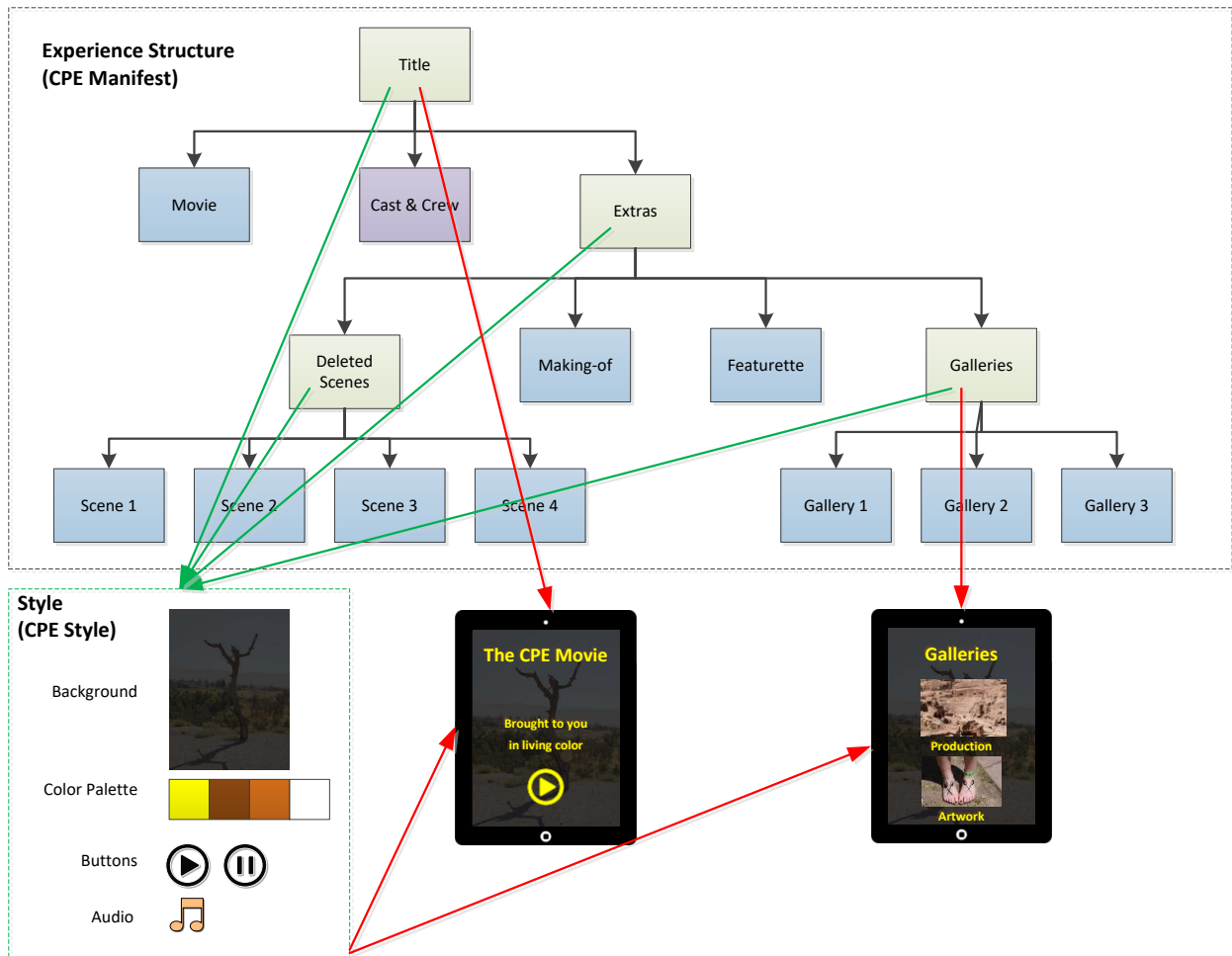
As much as iconic imagery is essential to the cinematic experience, text is a necessary part of the user experience. However, text is challenging to customize, mainly because it is difficult or expensive to render a specific font consistently at devices. The first level of customization comes in the color palette. Font rendered in a color consistent with the color scheme can help titles feel fresh and original.

The user app/interface designer, typically at the retailer, is free to take or ignore any of these style elements. However, we feel the designer will find that the inclusion of Node Style elements creates the best user experience.

2.2 Node Style and Experience Hierarchy

The Manifest file defines a hierarchy of experiences using the Experience and ExperienceChild elements. This hierarchy defines the structure and organization of the user interface. The type of each Experience and its location within the hierarchy will determine how Node Styles are applied.

The following illustrates how a Player combines the Experience hierarchy with CPE Style elements to create screens:



The Experiences that have style (i.e., not Audiovisual, Gallery or App Experiences) are mapped to one Node Style that contains background, color, buttons and audio. If there was more than one Node Style, different Experiences could map to different Node Styles.

The styling of Timed Event Sequences is supported indirectly. A Timed Event Sequence may be referenced by an Experience which is associated with a Node Style. Conversely, Experiences referenced by the Timed Events can have Node Styles. There are, however, currently no provisions for other Timed Event references to have Node Styles.

2.3 Adaptive and Responsive Design

The Appearance model is designed to support adaptive and responsive implementations. An *adaptive* user interface is able to support a wide range of viewing environments (e.g., 10' vs. laptop vs. mobile; HD vs. UHD). A *responsive* UI will respond to changes within a given environment (e.g., rotation of a tablet from portrait to landscape orientation).

Within a given Node Style, we attempt to be as responsive as possible. For example, the Background/Adaption element may be used to indicate how to “best fit” imagery to the available display screen. However, the Appearance model also allows selection of a Node Style based on orientation, device categories and resolution. Node Style selection is controlled via the ExperienceStyleMap. Section 3.1 describes in detail the structure and use of the ExperienceStyleMap while Section 6.6 covers the Background/Adaption element.

The given NodeStyle should work across the range of displays and devices it is applied to as indicated in the ExperienceStyleMap. Although not as comprehensive as HTML5/CSS, some the data structure provides some guidance regarding response to various environments.

Content authors and Profile Player implementers should be cautious not to make too many assumptions about displays. For example, text displayed on an 85” 4K TV might be readable while that same text is not readable on a 5” 4K mobile display. However, that mobile device could be casting to the 85” TV. This specification attempts to target the middle ground.

2.4 Putting it all together

A Player should use as much of the supplied Style information as it can while rendering backgrounds, text and graphical objects.

Following are some simple examples of backgrounds, colors and fonts.



3 CPE STYLE TOP LEVEL ELEMENTS

CPEStyleSet is a collection of elements that define a set of Node Styles and how they are to be assigned to the Experiences defined in a Manifest. There are three primary child elements:

- ExperienceStyleMap maps between ExperienceIDs and Node Styles (NodeStyleID) in the context of the target platform. ExperienceStyleMap instances are described in Section 3.1.
- NodeStyle instances define the style characteristics associated with one or more Experience elements. The NodeStyleSet-type is covered in Section 3.2.
- Theme instances define a set of common style characteristics that may be used by multiple NodeStyle instances thereby ensuring a consistent look-and-feel to the UI. The structure of the Theme element is covered in Section 4

| Element | Attribute | Definition | Value | Card. |
|-------------------------|---------------|--|---------------------------------|-------|
| CPEStyleSet-type | | | | |
| | CPEStyleSetID | Unique identifier for this Extras Menu | cpestyle:CPEStyleSetD-type | 0..1 |
| | updateNum | Version of this XML document. Initial release should be 1. This is a value assigned by the manifest creator that should only be incremented if a new version of manifest is released. If absent, 1 is to be assumed. | xs:integer | 0..1 |
| | specVersion | Version of spec to which the CPEStyleSet was authored. | xs:string | 0..1 |
| ExperienceStyleMap | | A map from ExperienceIDs to Node Styles. There should be one instance for each set of compatible display environments as defined in Compatibility. | cpestyle:ExperienceMenuMap-type | 1..n |
| NodeStyle | | A description of Node Styles. One instance is provided for unique device compatible set. | cpestyle:NodeStyleSet-type | 1..n |
| Theme | | Stylistic attributes defining color usage, font properties, and button sets. | cpestyle:Theme-type | 1..n |

3.1 Mapping Experiences to Styles: ExperienceStyleMap-type

The CPEStyleSet will contain one or more ExperienceStyleMap instances. Each instance identifies one or more Experiences, one or more NodeStyles, and, optionally, the conditions under which a specified NodeStyle is assigned to the referenced Experiences. The entire collection of ExperienceStyleMap instances allow a Player to determine what style to use for a given Experience. Therefore, each Experience must ultimately map to a NodeStyleID via one of the ExperienceStyleMap instances. All Experience IDs used in a Manifest must be included in exactly one map.

Multiple Experience elements may map to the same style. In this situation the Experience StyleMap instance will identify multiple instances of ExperienceID. This typically occurs when there are localized Experience instances. Note that if a style given style works for unrelated Experience elements, the Experience IDs for these elements could all be in the same map.

Multiple NodeStyleRef instances can be included with this set of ExperienceIDs. Each NodeStyleRef applies to display environment that are compatible with constraints defined by some combination of orientation, resolution and target device. This allows appropriate styles to be applied to different display environments.

| Element | Attribute | Definition | Value | Card. |
|--------------------------------|-----------|--|----------------------------|-------|
| ExperienceStyleMap-type | | | | |
| ExperienceID | | Experience ID that is described by the Node Style in NodeStyleID. Once instance exists for each ExperienceID that uses that Style. | manifest:ExperienceID-type | 1..n |
| NodeStyleRef | | Reference to the Node Style associated with the ExperienceIDs | cpestyle:NodeStyleRef-type | |

3.1.1 NodeStyleRef-type

A ExperienceStyleMap/NodeStyleRef instance identifies a single NodeStyle that the Player is to apply to all Experiences referenced by the ExperienceStyleMap. An optional set of constraint elements may be specified to limit the scope of Player environments in which the indicated NodeStyle is to be used. The constraints may consist of some combination of orientation, resolution and target device.

| Element | Attribute | Definition | Value | Card. |
|--------------------------|-----------|--|-----------------------------------|-------|
| NodeStyleRef-type | | | | |
| Orientation | | Orientation of the display to which the Node Style applies. If absent, then Node Style applies to any orientation. | xs:string | 0..1 |
| WidthPixelMax | | Maximum width in pixels of the intended display. If absent, then maximum width is unbounded. This corresponds with breakpoints in adaptive design. | xs:positiveInteger | 0..1 |
| DeviceTarget | | Target device or viewing experience for this XML document. One instance should exist for each Class that for which the Node Style applies. | cpestyle:CompatibilityDevice-type | 0..n |

Orientation is encoded as follows

- ‘Landscape – display is in landscape layout. This applies to 10’ displays and computer displays, as well as mobile devices when oriented in landscape mode (i.e., width ≥ height).
- ‘Portrait’ – display is in portrait layout. This applies typically to mobile devices in portrait mode (i.e., width < height)

3.1.2 CompatibilityDevice-type

| Element | Attribute | Definition | Value | Card. |
|---------------------------------|-----------|--|-----------|-------|
| CompatibilityDevice-type | | | | |
| Class | | General category of device or viewing experience for which this definition is intended | xs:string | |
| SubClass | | Further refinement of Class | xs:string | 0..n |

Class is encoded as follows:

- ‘Computer’ – general purpose computer such as a PC or Mac. Display could be built-in or external monitor. A laptop is considered to be a ‘Computer’, rather than a ‘Mobile’ device.

- ‘TV’ – any large screen display device. This refers to the “10 foot experience”.
- ‘Mobile’ – Smartphone, tablets and other hand-held/mobile devices. Although lines are blurred with tablets (phablets), a phone is typically 6” or smaller.

If the Node Style works across all devices of a particular Class, then SubClass need not be specified. However, if specific assets are required, SubClass can be used to provide additional detail on Class.

SubClass for Class of ‘Mobile’ can include

- ‘Phone’ – Smartphone devices. Although lines are blurred with tablets (phablets), a phone is typically 6” or smaller.
- ‘Tablet’ – Tablet devices, typically 7” or larger. Note that 6”-7” can use the Phone or Tablet moniker.

3.2 Node Style (Experience Appearance)

This element defines the various appearance aspects associated with a set of Experience elements.

Within this structure, Type and SubType provide categorization of the Node Style. Background defines background screens. ChildMenuStyle defines navigation. Other elements provide appearance description under certain circumstances.

3.2.1 NodeStyle-type

| Element | Attribute | Definition | Value | Card. |
|-----------------------|-------------|--|-----------------------------|-------|
| NodeStyle-type | | | | |
| | NodeStyleID | | | |
| Type | | General type of Node | xs:string | |
| SubType | | Specific type(s) of Node, when applicable. | xs:string | 0..n |
| ThemeID | | Reference to Theme for this Node Style | md:id-type | |
| Background | | Definition of background display and/or audio when applicable. | cpestyle:Background-type | 0..1 |
| Private | | For nonstandard additions to the Node Style. | Sequence 1..n xs:any##other | 0..1 |

Type is encoded as follows

- ‘Grouping’ – Style is of Grouping Style Type
- ‘Media’ – Style is of Media Style Type
- ‘App’ – Style is of App Style Type.
- ‘Special’ – Style is of Special Style Type.

4 THEMES

A Theme consists of some combination of color palette, a set of buttons images and/or optional colors.

4.1 Theme-type

Theme-type contains the elements of the Theme.

| Element | Attribute | Definition | Value | Card. |
|---------------------|-----------|--|----------------------------|-------|
| Theme-type | | | | |
| | ThemeID | | | |
| ColorPalette | | Palette of colors used in this Node Style. | cpestyle:ColorPalette-type | 0..1 |
| ButtonImageSet-type | | Button Image Set containing references to buttons and their variations | cpestyle:ButtonSet-type | 0..n |
| Fonts | | Font selection | cpestyle:Fonts-type | 0..1 |

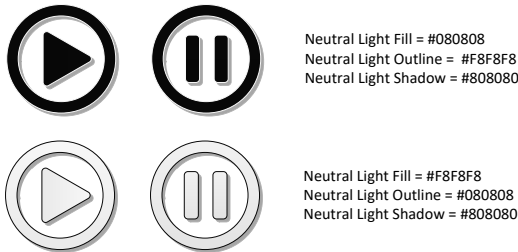
4.2 Color Palette

The Color Palette defines which colors are to be used for text and graphics when specific graphical elements are not provided. Multiple palettes are specified to cover various situations (e.g., when a graphic component is highlighted).

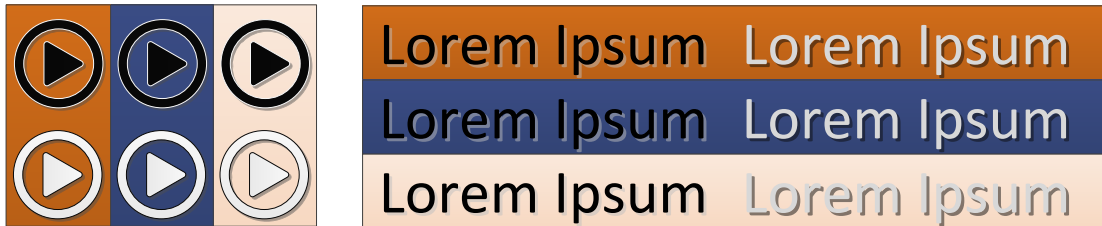
4.2.1 Neutral Palette

The Neutral Palette specifies whether object colors should be light or dark and is used whenever another palette is not specified in the XML or when a Player cannot handle a supplied color Palette. Players must provide a neutral palette for both light and dark objects (i.e., it is built into the Player).

Black, white and shades of grey are highly recommended to avoid clashing with colors. The following illustrates buttons and fonts rendered in representative dark and light palettes respectively:



Default objects, such as built-in buttons, should be drawn so that they can be used against light or dark backgrounds. The button and text examples above use outlines and shadows to stand out against different backgrounds.



4.2.2 Colored Palettes

Players should support coloring objects. At a minimum, outline and fill is supported. Shadow is recommended. The following illustration shows buttons rendered with an outline and fill color.



Subject to the UI design, when an object has focus or is rolled over, it is necessary to differentiate that object from other objects. Highlight colors are provided for the object in question. In the following example, the pause button uses the highlight colors.



Subject to UI design, some objects may be visible but deemphasized (defocused). For example, a button might be greyed out to indicate it is not valid option. Defocus colors are provided for such objects. The following example shows the pause button using defocus colors. In this case, the outline is the same as and the shadow is the same as the original's fill. However, the defocused button has no fill (i.e., transparent).



Button Defocus Outline = #8C4810
Button Defocus Shadow = #FFFF00

Following are text examples. Note that unless the player renders font fill different from outline, other methods must be used to show defocus (in this case, italic and slightly smaller)

Lorem Ipsum

Lorem Ipsum

Lorem Ipsum

4.2.3 Base, Highlight and Defocus

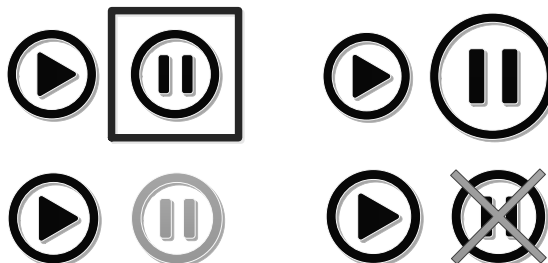
Some interfaces require objects to be highlighted or defocused for one reason or another. Highlight is used to so show are particular text or button has special status, such as focus or rollover. Some interfaces require objects to be visible, but to visually indicate they are not eligible for selection; often referred to as greyed out or defocused. We refer to this generally as defocus.

Highlight and defocus can be achieved through a variety of mechanism, such as changing size, changing weight, changing shape, changing color and outlining. The choice of method is at the discretion of the Player. Information is provided in the Style to provide the author's intent.

4.2.3.1 Note on Highlight and Defocus Design

Generally, when several buttons are visible, showing one button as different is sufficient to indicate highlight. However, when there are only two options, it can be difficult for the user to determine which is which (e.g., "Is the yellow or the blue button the one that's selected?"). It is strongly recommended that Player designers take this into account when designing highlight and defocus appearance. For example, increasing button size or outlining the button is a better indication than just color.

Following are some examples of showing highlight and defocus without color:



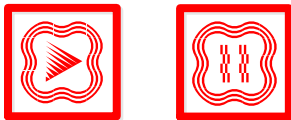
4.2.3.2 Highlight and Defocus of Neutral Palette Buttons and Text

As the Player provides buttons and text in the Neutral Palette, the Player must have the ability to modify these buttons for highlight or defocus.

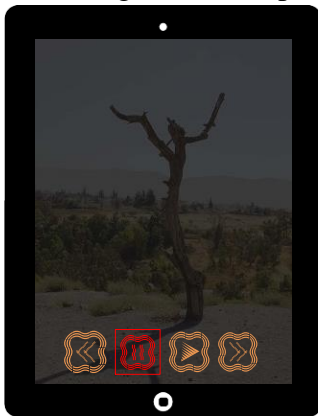
Colors are provided for highlight and defocus in NodeStyle/ColorPalette as defined in Section 4.2.3. ColorPalette/Base is intended for buttons neither highlighted nor defocused. ColorPalette/Highlight is intended for highlighted buttons. ColorPalette/Defocus is intended for defocused buttons.

4.2.3.3 Highlight and Defocus of Images

For image buttons, an alternate image is supplied. It is up to the author to make sure the effect works in the context of the other graphical elements. Following are examples of alternative images for the image buttons (possibly less attractive than the original):



Following is an example using alternate buttons. Focus is on Pause.



4.2.4 ColorPalette-type

| Element | Attribute | Definition | Value | Card. |
|-------------------|-----------|---|--|-------|
| ColorPalette-type | | | | |
| NeutralPalette | | Specifies whether object colors should be light or dark. This implies the use of built-in neutral default colors. | xs:string enumerations include 'light' and 'dark' | |

| | | | | |
|-----------|--|---|------------------------|--|
| Base | | Color set of base object; that is, an object that is neither highlighted nor defocused. | cpestyle:ColorHex-type | |
| Highlight | | Color set of highlighted object | cpestyle:ColorHex-type | |
| Defocus | | Color set of defocused object | cpestyle:ColorHex-type | |

NeutralPalette is encoded as follows

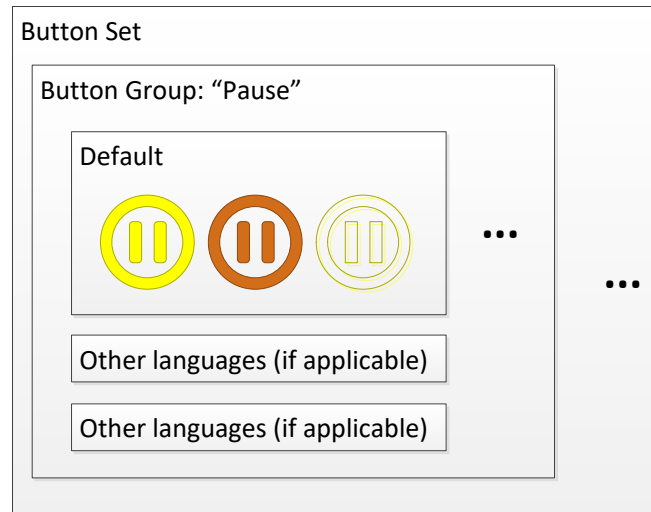
- ‘light’ – The light neutral color palette is to be used
- ‘dark’ – The dark neutral color palette is to be used.

4.2.5 ColorPalettInstance-type

| Element | Attribute | Definition | Value | Card. |
|---------------------------------|-----------|--|------------------------|-------|
| ColorPalettInstance-type | | | | |
| Outline | | Color to use on object outlines | cpestyle:ColorHex-type | |
| Fill | | Color to use for object fill. If absent, fill is transparent. | cpestyle:ColorHex-type | 0..1 |
| Shadow | | Color to use for object shadow. If absent, no shadow is applied. | cpestyle:ColorHex-type | 0..1 |

4.3 Button Image Set

A Button Image Set is a collection of all required buttons for a given Theme. A Button Image Set is comprised of Button Groups, with each Button Group containing all Buttons associated with a given Button Label (e.g., “Extras”). Within each Button Group, variations can be included for Base, Highlight and Defocus. Button Groups also support multiple languages. The following picture illustrates this grouping.



Note that when icons are used (i.e., no text) and localization is not necessary, a Button Group will contain only one entry. Localizations need only be applied to specific buttons where there is either language-specific text or regional iconography. For example, a Home button (🏠) may not need localization, but a Clip-N-Share button (clip/share) might.

4.3.1 ButtonSet-type

| Element | Attribute | Definition | Value | Card. |
|----------------|-----------|---------------------------|----------------------|-------|
| ButtonSet-type | | | | |
| Button | | Variations of one button. | cpestyle:Button-type | 1..n |

4.3.2 Button-type

| Element | Attribute | Definition | Value | Card. |
|-------------|-----------|---|----------------------------|-------|
| Button-type | | | | |
| | label | | | |
| Default | | Default button, independent of localization | cpestyle:ButtonImages-type | |

| | | | | |
|-----------|----------|--|----------------------------|------|
| | language | If present, the language associated with the default button. If absent, it is assumed that buttons do not have language-specific text. | xs:language | 0..1 |
| Localized | | Default button, independent of localization | cpestyle:ButtonImages-type | 0..n |
| | language | Language associated with buttons. | xs:language | |

4.3.3 ButtonImages-type

| Element | Attribute | Definition | Value | Card. |
|--------------------------|-----------|--|-----------------------|-------|
| ButtonImages-type | | | | |
| BaseImage | | Image to be used for button that is not highlighted or defocused | manifest:ImageID-type | |
| HighlightImage | | Image to be used for highlighted button | manifest:ImageID-type | |
| DefocusImage | | Image to be used for defocused button | manifest:ImageID-type | |

4.4 Fonts

The Theme fonts provide hints about the best looking fonts as well as details can that can be used by devices that are capable or more advanced font options.

4.4.1 Text Rendering Flexibility

Carefully thought out font face selection can greatly enhance the user experience, especially when the font face is consistent with the title treatment. However, typography is difficult and it is generally impractical to fully implement fonts. Information such as general font style and color palette is provided to the Player with the expectation that the Player will follow as much of that guidance as practical. More detailed information is provided for Players who can make use of those data.

The Player, however, is given considerable flexibility in rendering fonts, because rendering decisions are tightly coupled to Player UI design choices. For example, the Player knows how much space is available and can determine the correct font size for the text to fit.

Similarly, the Player knows how to best indicate highlight and defocus given the overall design of the UI.

4.4.2 Text Rendering

The Player should render text as closely as possible to the Theme’s definition. To do so, at a minimum, a Player should be able to

- Render Neutral ‘dark’ and neutral ‘light’ color schemes
- Render At least one each
 - Sans Serif
 - Serif
 - Monospace
- Show distinguishable sizes
- Render fonts in sizes that are distinguishable
- Render fonts with so that base (neither highlighted nor defocused), highlighted and defocused variations are distinguishable.

A Player should

- Support the ability to apply the Color Palette to all fonts.

4.4.3 Fonts-type

The Fonts-type provides the ability to specify fonts broadly, a specific font, and full CSS font definition.

Players need only support the FontGroup. However, broader font support is desired.

| Element | Attribute | Definition | Value | Card. |
|---------------|-----------|--|-----------|-------|
| Font-type | | | | |
| FontGroup | | General category for font | xs:string | |
| CSSFontFamily | | Font Family in accordance with the font-family property of CSS3 [CSS3FONT] | xs:string | 0..1 |

| | | | | |
|--------------------|--|---|-----------|------|
| CSS3FontProperties | | Font definition in accordance with CSS 3 font-related tags defined in [CSS3FONT]. Note: Not anticipated except for browser-based implementations. | xs:string | 0..1 |
|--------------------|--|---|-----------|------|

FontGroup is encoded as one of the following

- ‘serif’ – Serif font (e.g., Times Roman)
- ‘sansserif’ – Sans Serif font (e.g., Helvetica)
- ‘monospace’ – Monospace font (e.g., Courier New)

CSSFontFamily is encoded using the values of font-family property defined in [CSS3FONT]. For example, if the CSS property is “font-family: Verdana, Futura, Times;”, CSSFontFamily would be “Verdana, Futura, Times;”.

CSS3Font is included for advanced use. It is anticipated that only implementations based on web browser technology will make use of this element; at least in the near term. CSS3FontProperties includes one or more CSS3 font properties, in accordance with [CSS3FONT]. CSS3Font should use the ‘base’, ‘highlighted’ and ‘defocused’ class selectors to indicate their respective properties. For example,

```
.base {
  font-family: Helvetica, Verdana, sans-serif;
}
.highlighted {
  font-family: Helvetica, Verdana, sans-serif;
  font-weight: bold;
  font-stretch: expanded;
}
.defocused {
  font-family: Helvetica, Verdana, sans-serif;
  font-weight: lighter;
}
```

Color-related properties can override the Theme’s ColorPalette.

5 SELECTABLE OBJECTS: BUTTONS AND TEXT

For buttons, there are three levels of support. In increasing levels of flexibility, these are:

- **Built-in buttons:** buttons that are integral components of (i.e., built into) the Player application. These are, therefore, of the Player developer’s design.
- **Built-in button shapes:** Button templates that may be customized with colors from the Style Palette.
- **Image Buttons:** Buttons based on an image file identified in the CPE StyleSet file. Image buttons are, therefore, fully customizable. CPE Players may optionally support image buttons.

Similarly, for selectable text, there are

- Built-in font and neutral colors
- Fonts that can be rendered with colors from the Style Palette
- Custom fonts (limited support as described in Section 4.4)


5.1 Built-in Buttons

Built-in Buttons are buttons that are built into the Player application. These are, therefore, of the Player developer’s design.











The Player must provide a set of built-in buttons in two Neutral Palettes, referred to as “dark” and “light”. The “dark” buttons are best displayed over light backgrounds and conversely, “light” buttons are best displayed over dark backgrounds. Information on Neutral Palette can be found in Section 4.2.1. The Node Style specifies whether to use light or dark buttons. This is found in NodeStyle/ColorPalette/NeutralPalette as defined in Section 4.2.3.

All players should supply a set of built-in buttons. This document defines a minimal set. Button definitions, in addition to those listed in the table below, may be included in Interactivity Profiles.

The Button Label corresponds with ButtonGroup/@label.

| Button Label | Description | Comments | Example ¹ |
|--------------|-----------------------------------|--|---|
| Play | Start playback of audio or video. | This is distinct from the play or play/pause button in the audio/video player. |  |

¹ Some icons provided by <http://www.1001FreeDownloads.com>, who has our appreciation.

| | | | |
|---------------|--|--|---|
| Extras | Go to Extras page | |  |
| Featured | Go to featured extras page | |  |
| Share | Share on social networking | |  |
| ShareClip | Share a clip on social networking | | |
| Search | Search | |  |
| Help | Help | |  |
| Cast and Crew | Go to Cast and Crew page (or contextual cast and crew) | |  |
| Back | Go back | |  |
| Exit | Exit (or home) | |  |
| Bookmark | Bookmark location | |  |
| Favorites | Favorite | |  |

Note that player buttons such as pause, skip and fast-forward/backward are not included in this set.

5.2 Built-in Button Shapes

CPE Players should support colored buttons that are created from templates. Node Style will include a color for the outline and optionally one for fill. If fill is not provided, the button's fill areas are transparent. A shadow color may also be provided. The position of the shadow is at the Player's discretion. Note that offset is dependent on factors such as button size, screen size and viewing distance so this is best left to the Player. Information on Colored Palettes can be found in Section 4.2.2.

If a Color Palette is provided, it will be found in NodeStyle/ColorPalette as defined in Section 4.2.3.

5.3 Image Buttons

CPE Players may optionally support buttons that use images defined by the Theme, rather than the Player. Images are in PNG format and transparency must be supported. Image Buttons are referenced as part of a Theme via NodeStyle/ThemeID. Themes are described in Section 4.

These rather unattractive images are used for illustration purposes:

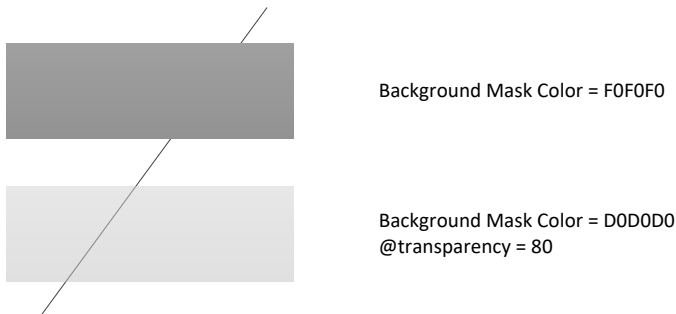


5.4 Built-in Fonts

5.5 Button and Label backgrounds

If supplied buttons are being used (i.e., color or image) and background images and/or colors are not supported, it might be necessary to provide a background behind the buttons.

These can be specified by a color or a color with alpha. Following are examples of button backgrounds with and without transparency. Note that they look much similar against the actual background. The line is included just to show transparency.



The size of the button background must be larger than the button, but the exact design is up to the Player. For example, the Player designer may wish to extend the button background to the edge of the window.

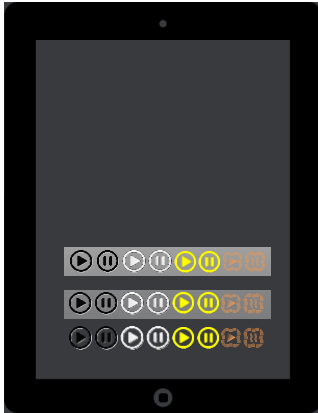
5.6 Compositing

The most background element is the color, image or video background. The next layer is the button background, if required. The final layer is the buttons. Note that transparency must be supported for all graphical elements except the background.

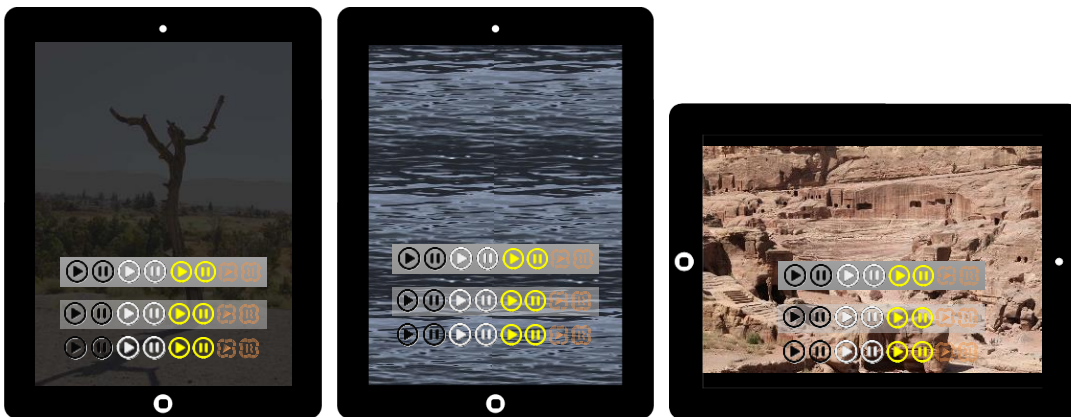
Each example shows a combination of built-in buttons, colored buttons and image buttons. These buttons are shown over solid button backgrounds, transparent button backgrounds and no background.

Note that as Players have different capabilities, it is important that designers consider different combinations.

The following example shows a solid color background



The following examples show image backgrounds, both full images and tiled images. A video background would look similar.



Note that the colored and image buttons were designed for the first image and do not translate well to the other images. However, with the proper selection of light or dark buttons and/or the use of button backgrounds all buttons are visible.

6 BACKGROUND

Menus, Galleries and some applications can have a background consisting of some imagery (image or video), audio and certain menu elements.

Within this design, enough flexibility it included to deal with a particular range of devices. However, as not all display layouts can be addressed with one set of assets, multiple instances of background definitions can be included. For example, one might be provided for portrait and another for landscape.

As discussed in Section 2.3, the range of display environment can be limited by `NodeStyleRef`. Using `NodeStyleRef`, the CPE Player picks the best match for the display. Once that selection is made, the rest of this section describes how to display.

6.1 Combining Elements into a Background

Backgrounds may contain at most one visual and one audio element.

The visual element can be a color, an image or a video. A background color can be in lieu of background image or video; or in conjunction with a video or image. If background color is provided in addition to an image or video, the background color appears where the image or video does not. If no background color is provided, the player may provide on at its own discretion.

If video is playing audio should come from the video. `AudioLoop` should be provided for devices that do not support video.

Audio and video are assumed to be looped.

The following combinations have defined behavior:

- Image: Image is displayed
- Image + audio: Image is displayed, audio is played.
- Video without audio + Image: Video is played if possible. If not, Image is displayed. Video playing with a distinct audio track is not allowed.
- Video with audio + Image: If possible video, including audio, is played. If video is not possible then image is displayed and audio portion of audiovisual is played.

6.2 Background Assets

Within the style, one or more of the following will be provided: Color, image, audio and Video (possibly with audio).

The CPE Player should display the best combination it is capable of displaying. Generally, the hierarchy for visuals is video, image and color. Regarding audio, best is the audio

that comes with video (assuming video is played), then looped audio. If audio is provided, it should be played.

6.2.1 Background-type

| Element | Attribute | Definition | Value | Card. |
|------------------------|-----------|--|-------------------------------|-------|
| Background-type | | | | |
| | looping | Audio and/or video will loop (repeat from beginning once the end is reached). | xs:boolean | |
| Color | | Background color either in lieu of images or in addition to images. | cpestyle:ColorHex-type | 0..1 |
| Image | | Image to be used in background | cpestyle:BackgroundImage-type | 0..1 |
| Video | | Video to be used in background | cpestyle:BackgroundVideo-type | 0..1 |
| AudioLoop | | Audio, possibly looped, to be played in background. | cpestyle:BackgroundAudio-type | 0..1 |
| Adaptation | | Information on how to adapt the background elements to display environment (change of size or aspect ratio). | cpestyle:Adaptation-type | 0..1 |

Looped audio and video should be authored for looping. Typically, the beginning and end of the audio should have some period of silence. Video should also loop without an unpleasant transition.

6.2.2 Background Color

The lowest common denominator for background is color. A CPE Player should only display color if neither a video or image is provided.

The following example uses an image color of #393A3E. This color is consistent with image backgrounds in other examples. The use of consistent colors ensures overlays, such as buttons, will look correct whether overlaid on images or on simple backgrounds.



Note that when anything other than a uniform color is desired (e.g., a gradient), an image should be used.

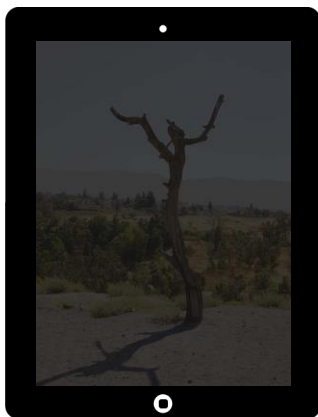
Background color is also used for letter and pillar boxes when an image does not fit the window.



6.3 Background Image

Background images can be provided for given Node Style. These images should be the same except for localization. To support localization, image references are to Picture Groups (which support multiple images that differ only in localization).

Note that the following example has a dark, low contrast image. This makes brighter overlays more readable.



6.3.1 BackgroundImage-type

BackgroundImage defines a single image or images to be played in sequence.

| Element | Attribute | Definition | Value | Card. |
|-----------------------------|-----------|---|-------------------------|----------|
| BackgroundImage-type | | | | |
| Inherit | | Properties are inherited from the parent Experience. | xs:boolean | (choice) |
| PictureGroupID | | Reference to a Picture Group that contains the image. | cpestyle:PictureGroupID | |
| Slideshow | | Reference to a slide show (rotating images). | cpestyle:Slideshow-type | |

6.3.1.1 SlideShow-type

Defines a series of images to be played either in sequence or shuffled. All images are played from a PictureGroup. If the Player cannot play multiple images, just the first should be used.

| Element | Attribute | Definition | Value | Card. |
|-----------------------|-----------|---|-------------------------|-------|
| SlideShow-type | | | | |
| PictureGroupID | | Reference to a Picture Group that contains the image. | cpestyle:PictureGroupID | |
| Shuffle | | If 'true' slides are played randomly. If 'false' or absent, slides are played in sequence. | xs:boolean | 0..1 |
| Duration | | How long to hold each image before switching | xs:duration | 0..1 |
| Transition | | Method of transitioning between images. If absent, slides change without transition. If absent, 'cut' is assumed. | xs:string | 0..1 |

Transition is encoded as follows:

- 'cut' – no transition (default)
- 'fade' – fade to new image
- 'kenburns' – Ken Burns effect style

6.4 Background Video

Background video is laid out just like images.

Background video is looped. There is an option to loop an initial sequence, then loop the remainder.



If video is inherited, it should continue playing throughout navigation.

If a video background is played, audio must come from the video; not from BackgroundAudio.

If a subset of a Presentation is desired, reference a Playable Sequence with a single Clip whose EntryTimecode and ExitTimecode delimit the desired subset.

6.4.1 BackgroundVideo-type

| Element | Attribute | Definition | Value | Card. |
|-----------------------------|-----------|--|----------------------------------|----------|
| BackgroundVideo-type | | | | |
| Inherit | | Properties are inherited from the parent Experience. | xs:boolean | (choice) |
| PresentationID | | Reference to a Presentation of the background video. | manifest:PresentationID-type | |
| PlayableSequenceID | | Reference to a Playable Sequence that defines the video background. Typically, this is only used for subsets of a longer Presentation defined in a Clip. | manifest:PlayableSequenceID-type | |
| LoopTimecode | | Timecode for audiovisual looping. Must be greater than zero and less than the end of the Presentation or Clip. Exclude if zero (i.e., no looping). | manifest:Timecode-type | 0..1 |

6.5 Background Audio

If the video has audio, it should be played with the video. If not, audio should be played as referenced.

Background audio is looped. There is an option to loop an initial sequence, then loop the remainder.



If audio is inherited, it should continue playing throughout navigation.

Note that switching audio can be distracting and should be used judiciously.

6.5.1 BackgroundAudio-type

This type defines an audio loop. If it is absent, it is assumed that audio is inherited from the parent Experience. If the Player navigates to an Experience with an identical AudioLoop or no AudioLoop, the Player should continue playing the clip (i.e., without restarting or with interruption).

| Element | Attribute | Definition | Value | Card. |
|-----------------------------|-----------|--|----------------------------|-------|
| BackgroundAudio-type | | | | |
| AudioTrackID | | See manifest:AudioClipRef-type | manifest:AudioTrackID-type | |
| EntryPointTimecode | | See manifest:AudioClipRef-type | manifest:Timecode-type | 0..1 |
| ExitPointTimecode | | See manifest:AudioClipRef-type | manifest:Timecode-type | 0..1 |
| LoopTimecode | | Timecode for audio looping. Must be greater than EntryPointTimecode and less than ExitPointTimecode. Exclude if equal to EntryPointTimecode. | manifest:Timecode-type | 0..1 |

6.6 Adaptation

Unless the Node Style is targeted for a particular device, the image will need to be scaled and/or cropped to match the display. The following defines the Scale Method. The ScaleMethod element is encoded using the following values (in quotes):

- “BestFit” – Scale the image to the display so the image fills the display and is fully displayed along one axis. That is, the image should fit exactly either

horizontally or vertically; while some of the image might be cropped in the other dimension.

- “Full” – scale the image to ensure the entire image is visible. Pillarboxes or Letterboxes are filled with the background color.
- ‘Tiled’ – Image is tiled. Tiling always begins in orientation with the text. That is, tiling is left-to-right for English, and right for right-to-left languages such as Hebrew. Vertical written text should be left-to-right or right-to-left depending on columns layout.

The aspect ratio must be maintained when cropping or scaling an image. Scaling asymmetrically (i.e., squashing horizontally or vertically) is prohibited.

Following are examples of scaled images. The first is Best Fit and the other is Full. Note that the entire image is show for illustration but would be cropped to the dimension of the device. Note the pillarboxes in the fit image.

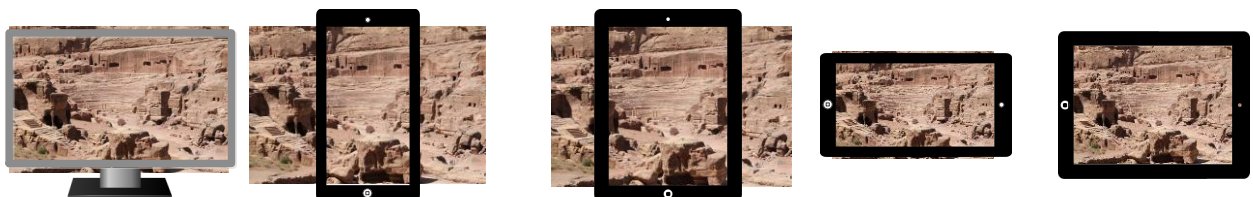


Unless the scaled image exactly matches the display, positioning is required. Positioning Methods are as follows. The PositionMethod element is encoded using the following values (in quotes):

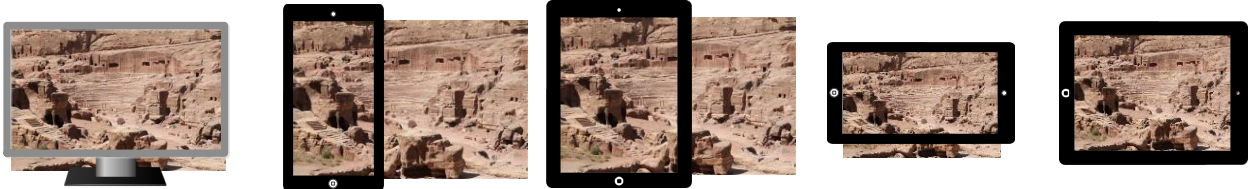
- ‘upperleft’ – position the upper left corner of the image in the upper left corner of the screen
- ‘upperright’, ‘lowerleft’, ‘lowerright’ – same as upperleft, but positioned at the upper right, lower left and lower right respectively.
- ‘centered’ – The center point of the image is at the center point of the screen (give or take a pixel).

Following are layout examples:

Best Fit, Centered



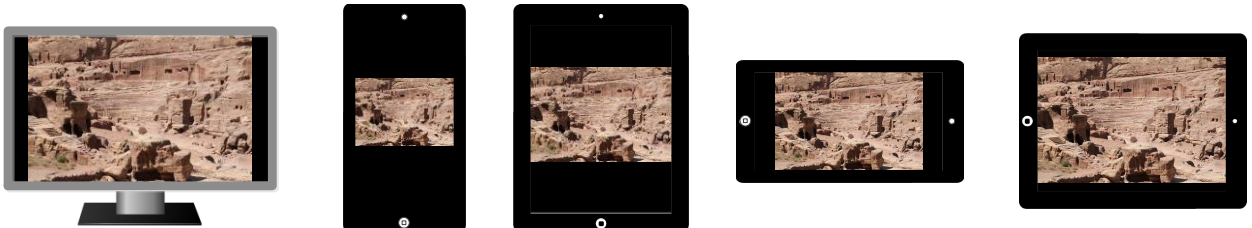
Best Fit, Upper Left



Full, Upper Left

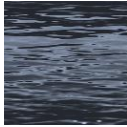


Full Centered



Tiled

This image:



Tiles as follows:



6.6.1 Adaptation-type

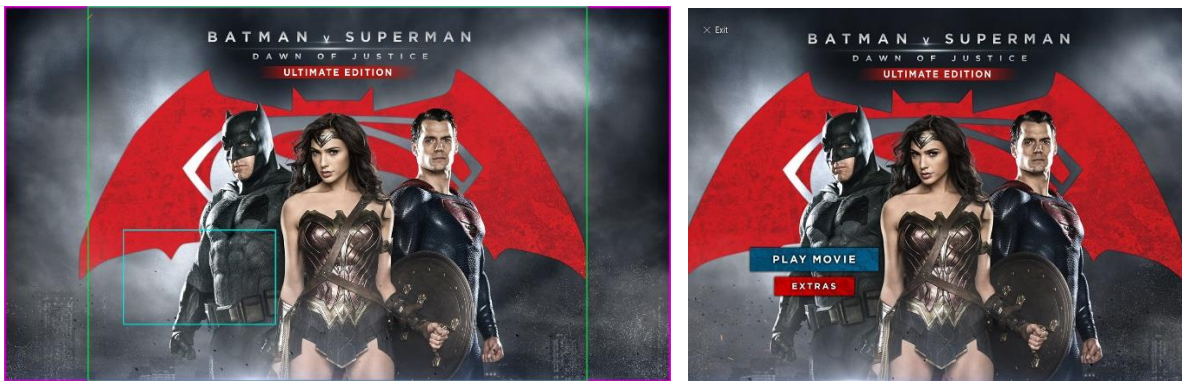
| Element | Attribute | Definition | Value | Card. |
|------------------------|-----------|--|----------------------------|-------|
| Background-type | | | | |
| ScaleMethod | | Specified the how images and videos are rendered into the viewport. Values are as defined above. | xs:string | 0..1 |
| PositioningMethod | | Positioning Method. Values are as defined above. | xs:string | 0..1 |
| FillColor | | If image or video doesn't cover background, what color should be used to fill letter- or pillar-box. Default is black (#000000). | cpestyle:ColorHex-type | 0..1 |
| SafeArea | | Defines area that contains essential elements of the image. | cpestyle:ImageSubArea-type | 0..n |

6.7 Overlay Areas

An Overlay Area is a subset of an image that can be overlaid with graphical objects without compromising the background image. This is especially important when background art has images of people.

This is designed to handle cases, such as start screens, where the UI background image is primary, overlaid with a small number of menu options. Overlay Areas define areas where to place these objects.

The following examples show a 16:9 image with a 4:3 safe area outlined in green. The Overlay Area is show in green. The final image is cropped with buttons overlaid.



Overlay areas can be tagged for specific use. For example, at area design for a menu might be tagged 'menu' by setting @tag="menu".

Overlay areas can be ordered so that preferred areas are used before less preferred areas. Where possible, Overlay Areas with a higher priority should be used before Overlay Areas with lower priority.

Overlay Areas are constrained as follows:

- Safe area and overlay area cannot overlap
- Overlay area can cover the entire image, excepting the safe area
- Overlay area is consistent with the "Theme". In particular, text or graphics rendered using the ColorPalette or ButtonImageSet will look correct. This is for particular applications and a ButtonImageSet will define the buttons in question.

6.7.1 BackgroundOverlayArea-type

| Element | Attribute | Definition | Value | Card. |
|-----------------------------------|-----------|---|--|-------|
| BackgroundOverlayArea-type | | | cpestyle:ImageSubArea-type (by extension) | |
| | tag | A label for the intended use of this overlay area. | xs:string | 0..1 |
| | priority | Relative priority for use. A value means a higher priority. | xs:integer | 0..1 |