Notes

The purpose of this document is to instruct a user in how to author a Note Script.

**CCV APP 2.0** contains a special “Notes” section of the app. Rather than being a static UI, the “Notes” section is dynamically created by reading a Note Script.

Note Scripts utilize XML and a XAML-like syntax, so a familiarity with XML/XAML will allow a user to quickly begin creating notes.

Useful Links:

XML: <http://www.w3schools.com/xml/xml_whatis.asp>

XAML: <http://msdn.microsoft.com/en-us/library/ms752059(v=vs.110).aspx>

Table of Contents

Note Script Overview 1

Script Basics 2

Designing a Note Script 2

Style Sheet 4

Assigning a Style Sheet to a Note Script 4

Controls and Behaviors 5

Positioning, Padding and Child Controls 6

Positioning 6

Padding 7

Child Controls 8

Control Reference 9

StyleSheet 9

StackPanel 10

Paragraph 10

Header 11

Quote 11

RevealBox 11

TextInput 12

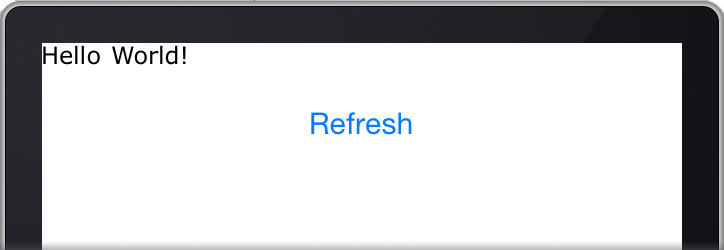
## Note Script Overview

The Note Script is an XML document utilizing XAML-like syntax that instructs **CCV APP 2.0** in how to build the UI to be seen in its “Notes” section.

Consider the following Note Script:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> |
| <Note StyleSheet="http://www.hosting.com/style.xml"> |
|  |
| <Paragraph>Hello World!</Paragraph> |
|  |
| </Note> |

This would result in a native UI page that simply displayed “Hello World” in the upper left corner of the screen, as seen in the following figure.



The specifics will be detailed below, but this illustrates the concept that the Note Script is parsed and a dynamic UI is built from it.

## Script Basics

To begin writing a Note Script, there are several requirements. These requirements can be seen in the sample above, repeated again here.

|  |
| --- |
| **1. <?xml version="1.0" encoding="UTF-8"?>** |
| **2. <Note StyleSheet="http://www.hosting.com/style.xml">** |
|  |
| **3. </Note>** |

1. The first requirement is simply a standard XML tag stating that the document will be in the XML format.

2. The second requirement is the root element <Note>. The Note Script contents will all be within the <Note> element.

The <Note> element contains one required Attribute, ‘StyleSheet’. This

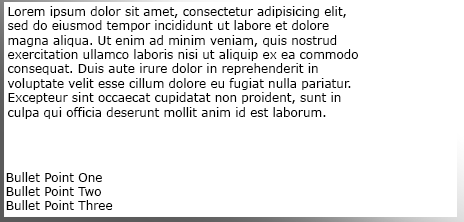
sheet describes the default style for controls, and is required. It is discussed later in this document.

3. The last declaration in a Note Script should be the closing </Note> element tag.

## Designing a Note Script

Note Scripts can be designed similar to how an HTML document with CSS is designed. (Note that neither HTML nor CSS are supported.) Using built-in controls and containers, a user can layout a page that matches a desired design.

For example, if the design required an introductory paragraph followed by three bullet-points, a **Stack Panel**, followed by four **Paragraphs** would be used. (Don’t worry about what a Stack Panel is just yet.)

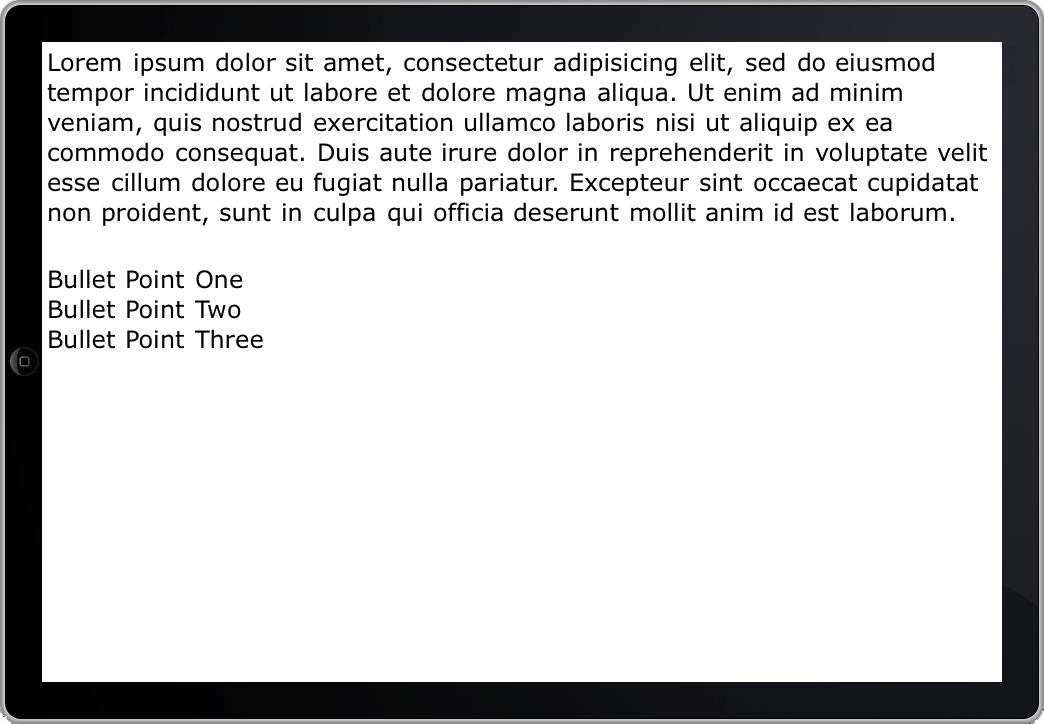


This example document is NOT a Note. The goal is to write a Note Script that will create a note that matches its layout.

Looking at the design above, the Note Script below is authored.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> |
| <Note StyleSheet="http://www.hosting.com/style.xml"> |
| <StackPanel> |
| <Paragraph>Lorem ipsum dolor sit amet, consectetur adipisicing elit,  sed do eiusmod tempor incididunt ut labore et dolore  magna aliqua. Ut enim ad minim veniam, quis nostrud  exercitation ullamco laboris nisi ut aliquip ex ea commodo  consequat. Duis aute irure dolor in reprehenderit in  voluptate velit esse cillum dolore eu fugiat nulla pariatur.  Excepteur sint occaecat cupidatat non proident, sunt in  culpa qui officia deserunt mollit anim id est laborum.</Paragraph>  <Paragraph>Bullet Point One</Paragraph>  <Paragraph>Bullet Point Two</Paragraph>  <Paragraph>Bullet Point Three</Paragraph> |
| </StackPanel> |
| </Note> |

When **CCV APP 2.0** runs and parses the above Note Script, the result would be a Note similar to the image below.



Notice that it very closely matches the original design requirement.

## Style Sheet

Now that the Note Script concept has been established, it is critical to understand the various components that create the design.

The **StyleSheet** attribute of the **<Note>** element defines the Style sheet to use.

A style sheet is also an XML with XAML-like syntax, and describes the default behavior of various controls. It is important to note that any control setting its own behaviors will override these defaults.

Here is an example style sheet:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> |
| <Styles> |
| <Note BackgroundColor="0xFFFFFFFF"/> |
| <Text FontName="Verdana" FontSize="12" FontColor="0x000000FF"/> |
| </Styles> |

This style sheet specifies that the background color of the note should be black**,**

text should be white, and it should use the Verdana font and a point size of 12.

A complete list of supported Elements and Attributes can be found in the [**Control Reference**](#_Control_Reference) section.

### Assigning a Style Sheet to a Note Script

Style sheets are assigned to a Note Script via the StyleSheet attribute in the <Note> element

as follows:

|  |
| --- |
| <Note StyleSheet=”URL\_TO\_STYLE\_SHEET”> |

Style Sheets must be referenced via URL and cannot be embedded in the Note Script.

## Controls and Behaviors

Controls (and containers) are at the heart of Note Script, and allow content to be defined and positioned.

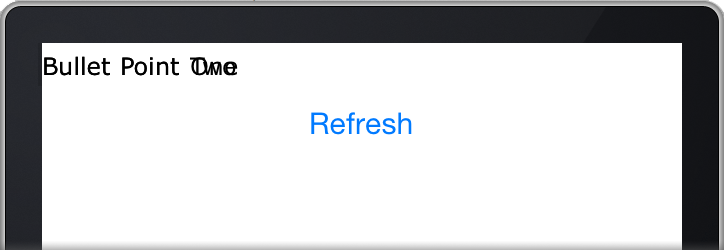
A complete list of controls and their supported Attributes and Elements can be found in the [**Control Reference**](#_Control_Reference_1)**.**

The <Note> element itself represents a blank canvas, with no placement logic built in. This means if two <Paragraph> controls were placed with no positioning, they would appear on top of each other in the upper-left (default) corner of the Note.

Example:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> |
| <Note StyleSheet="http://www.hosting.com/style.xml"> |
| <Paragraph>Bullet Point One</Paragraph>  <Paragraph>Bullet Point Two</Paragraph>  </Note> |

The above Note Script would result in the following Note being generated.



One solution would be to set positions for each <Paragraph> control as follows:

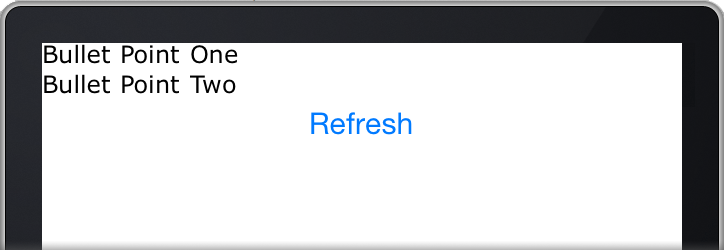
|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> |
| <Note StyleSheet="http://www.hosting.com/style.xml"> |
| <Paragraph **Top=”0”**>Bullet Point One</Paragraph>  <Paragraph **Top=”25**”>Bullet Point Two</Paragraph>  </Note> |

This would result in the second <Paragraph> being below the first. However, doing this would require setting every control’s position explicitly and would obviously be tedious and time consuming.

A better way is to use a container control that supports positioning. One such control is known as the <StackPanel>. This control places child controls in a vertical stack. A complete list of container controls can be found in the [**Control Reference**](#_Control_Reference_2).

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> |
| <Note StyleSheet="http://www.hosting.com/style.xml"> |
| <StackPanel> |
| <Paragraph>Bullet Point One</Paragraph>  <Paragraph>Bullet Point Two</Paragraph> |
| </StackPanel> |
| </Note> |

The above Note Script is similar to the previous example, except it adds a stack. The resulting Note would look as follows.



## Positioning, Padding and Child Controls

When designing anything but the most basic Note, it will quickly become necessary to position controls, set margins and place controls within controls (child controls.)

**CCV App 2.0** offers multiple ways to accomplish these things. Each will be covered here.

### Positioning

Positioning a control is accomplished with the Attributes Left and Top. For a full list of attributes supported by each control, see the [**Control Reference**](#_Control_Reference_3).

A position can be specified in either pixels or percentage of parent width. Positioning is always relative to the control’s parent.

**Pixels**

Consider a <Paragraph> that is a child of <Note>.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> |
| <Note StyleSheet="http://www.hosting.com/style.xml"> |
| <Paragraph **Left=”25”**>Some Example Text</Paragraph>  </Note> |

The <Paragraph> would be placed 25 pixels from the left edge of its parent.

Now consider the same <Paragraph> as a child of a <StackPanel> that has been placed 25 pixels to the left of its parent’s edge.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> |
| <Note StyleSheet="http://www.hosting.com/style.xml">  <StackPanel **Left=”25”**> |
| <Paragraph **Left=”25”**>Some Example Text</Paragraph>  </StackPanel>  </Note> |

This <Paragraph> would be placed 25 pixels from the left edge its parent.

However, because <StackPanel> is 25 pixels from *its*parent, this would result in the <Paragaph> being a total of 50 pixels from the left edge of the Note.

**Percentage**

Another way to position controls is with percentages. Percentages allow “relative” layouts and work better when a Note will be displayed on devices with varying screen sizes.

Percentages are denoted with a ‘%’ symbol and are in terms of the parent’s width or height.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> |
| <Note StyleSheet="http://www.hosting.com/style.xml"> |
| <Paragraph **Left=”10%”**>Some Example Text</Paragraph>  </Note> |

This <Paragraph> would be placed 10% of the parent’s width to the left. For example, if the parent were 100 pixels wide, this would result in <Paragraph> being 10 pixels from its parent’s left edge.

### Padding

Padding allows a control to state “No content should be closer than *n* units” from a given edge. For example:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> |
| <Note StyleSheet="http://www.hosting.com/style.xml">  <StackPanel **PaddingLeft=”10”**> |
| <Paragraph>Some Example Text</Paragraph>  </StackPanel>  </Note> |

Normally the <Paragraph> would be at the edge of <StackPanel>. However, because the padding specifies 10 pixels, the <Paragraph> will be 10 pixels from the left edge of the <StackPanel>.

**Padding reduces the available width of child controls.** This means that if the <StackPanel> in the above example were 100 pixels wide, its child controls would only have 90 pixels to fit within. This is desirable behavior as it prevents controls from “bleeding” outside their boundaries.

**Pixels vs Percentage**

As with positioning, padding can be specified in pixels or percentage.

### Child Controls

The Note Script is designed to be intuitive and consistent with HTML / XML behavior. This means that if a control is inside another control, it is considered a child of the higher-level control.

There are limits to which controls support children. For a complete listing, see the [**Control Reference**](#_Control_Reference_4).

**Inheritence**

Child controls inherit their parent’s attributes. For example, if a parent control specifies a particular Font, all children will use that Font unless specifically overridden. Consider the following example:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> |
| <Note StyleSheet="http://www.hosting.com/style.xml">  <StackPanel **FontName=”Verdana”**> |
| <Paragraph>Some Example Text</Paragraph>  <Paragraph **FontName=”Helvetica”**>Some Example Text</Paragraph>  </StackPanel>  </Note> |

The first <Paragraph> would use Verdana, as it is a child of <StackPanel> and does not specify a Font to use.

The second <Paragraph> would use Helvetica because it specifically requested it.

If an attribute is not specified in a control or any of its parents, it defaults to the StyleSheet.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> |
| <Note StyleSheet="http://www.hosting.com/style.xml">  <StackPanel> |
| <Paragraph>Some Example Text</Paragraph>  <Paragraph>Some Example Text</Paragraph>  </StackPanel>  </Note> |

In this example, both <Paragraph> controls would use the values set to the <Paragraph> in StyleSheet.

## Control Reference

### StyleSheet

Summary: An XML file containing default attributes for various controls.

Style sheets consist only of root-level elements with attributes specifying defaults.

Following are the supported elements and their supported attributes.

Note

BackgroundColor

Paragraph

PaddingLeft, PaddingTop, PaddingRight, PaddingBottom

BackgroundColor

FontName, FontSize, FontColor

RevealBox

BackgroundColor

FontName, FontSize, FontColor

TextInput

BackgroundColor

FontName, FontSize, FontColor

Quote

BackgroundColor

FontName, FontSize, FontColor

Header

BackgroundColor

FontName, FontSize, FontColor

Text

BackgroundColor

FontName, FontSize, FontColor

### StackPanel

Summary: Container control that places children in a vertical stack.

Supported Attributes:

*Positioning*

Left, Top

*Sizing*

Width

*Spacing & Alignment*

PaddingLeft, PaddingTop, PaddingRight, PaddingBottom

**Note:** Children **do not** inherit padding values.

Alignment, ChildAlignment

*Appearance*

FontName, FontSize, FontColor

BackgroundColor

Debug

Supported Elements:

StackPanel

Paragraph

Header

RevealBox

Quote

TextInput

### Paragraph

Summary: Container control that places text in a paragraph format with word wrapping.

Supported Attributes:

*Positioning*

Left, Top

*Sizing*

Width

*Spacing & Alignment*

PaddingLeft, PaddingTop, PaddingRight, PaddingBottom

**Note:** Children **do not** inherit padding values.

Alignment, ChildAlignment

*Appearance*

FontName, FontSize, FontColor

BackgroundColor

Debug

Supported Elements:

RevealBox

### Header

Summary: Displays a title, speaker and date with appropriate header formatting.

Supported Attributes:

*Positioning*

Left, Top

*Spacing & Alignment*

Alignment

*Appearance*

FontName, FontSize, FontColor

BackgroundColor

Debug

Supported Elements:

Title

Speaker

Date

### Quote

Summary: Displays quoted text with word wrapping and a citation.

Supported Attributes:

*Custom*

Citation

*Positioning*

Left, Top

*Sizing*

Width

*Spacing & Alignment*

PaddingLeft, PaddingTop, PaddingRight, PaddingBottom

Alignment

*Apperance*

FontName, FontSize, FontColor

BackgroundColor

Debug

Supported Elements

None

### RevealBox

Summary: Displays text that is hidden until a user taps on it.

Supported Attributes:

*Content*

Text

*Positioning*

Left, Top

*Spacing & Alignment*

Alignment

*Appearance*

FontName, FontSize, FontColor

BackgroundColor

Debug

Supported Elements:

Text

### TextInput

Summary: Displays an interactive text field with placeholder text.

Supported Alignment:

*Content*

Text

*Positioning*

Left, Top

*Sizing*

Width

*Spacing & Alignment*

Alignment

*Appearance*

FontName, FontSize, FontColor

BackgroundColor

Debug

Supported Elements:

PlaceHolder

### Attribute Values

Attributes support only certain types of values. Each attribute and its supported value are listed here.

**Positioning**

Left, Top – Numeric pixel or percentage value.

Ex: Left=”50” or Left=”50%”

**Sizing**

Width, Height – Numeric pixel or percentage value.

Ex: Width=”50” or Width=”50%”

**Spacing & Alignment**

Alignment, ChildAlignment – “Left”, “Center” or “Right”

Ex: Alignment=”Center”

PaddingLeft, PaddingTop, PaddingRight, PaddingBottom – Numeric pixel or percentage value.

Ex: PaddingLeft=”50” or PaddingLeft=”50%”

**Appearance**

FontName – Legal Font Name.

Ex: FontName=”Verdana”

FontSize – Point size for the font.

Ex: FontSize=”16”

FontColor – RGBA color for in Hexidecimal format.

Ex: FontColor=”#RRGGBBAA”

BackgroundColor - RGBA color in Hexidecimal format.

Ex: BackgroundColor=”#RRGGBBAA”

Debug – True or False value

Ex: Debug=”True”