UI DESIGN AND INTERACTION GUIDE

FOR WINDOWS® PHONE 7 SERIES

Contents

UI Design and Interaction Guide for Windows Phone 7 Series

The Windows® Phone 7 Series CTP User Interface (UI) is based off a Windows Phone design system internally codenamed "Metro." The Metro design principles center on a look that uses type to echo the visual language of airport and metro system signage. The goal is to clearly direct end users to the content they want. Metro interfaces are supposed to embody harmonious, functional, and attractive visual elements. Ideally, good UI design should encourage playful exploration when interacting with the application and people should feel a sense of wonder and excitement. A clear, straightforward design not only makes an application legible, it encourages usage. This guide will provide design knowledge and fundamentals for this type of UI development. We highly recommend that developers adopt the Metro design style whenever possible. Although requirements may vary based on the application, paralleling this experience will create a more consistent, fluid UI experience from the custom and built-in application view.

This guide will also detail several possible methods of interaction that can be used by a Windows Phone 7 Series CTP application, including standard input, functionality within the UI framework, and the Metro-themed Silverlight® and system-based controls. By understanding and incorporating key design concepts and considerations within these areas, you can craft your application to provide a better end user experience. You will also have a deeper understanding of the number of different hardware and software interaction elements that are available to developers in Windows Phone 7 Series CTP. Diverging from the Windows Phone 7 Series CTP interaction model is generally not allowed. However, there are a few exceptions where the UI behavior is different as application requirements vary.

Select platform features that developers can leverage include the following:

- ✓ **Hardware** - The Windows Phone hardware features a large WVGA multi-touch screen and incorporates components like GPS functionality, and sensors such as an accelerometer. User navigation and input on this hardware can be completed by way of "gestures"—finger-based movements upon the capacitive touch screen. Developers can use these components for their applications, creating an enhanced experience for the end user.
- ✓ **UI Framework** - The Windows Phone 7 Series CTP UI framework is modeled to provide an optimal viewing experience for the end user. For example, an application can switch between portrait and landscape modes for better viewing conditions. This guide will discuss the frame page navigation model used in Windows Phone 7 Series CTP, and the ability to theme your applications to personalize the visual elements.
- ✓ **Silverlight** - The Silverlight® UI framework delivered on Windows Phone 7 Series CTP is the next step in the evolution of Silverlight and enables a new class of mobile design experiences. Silverlight harnesses the power of .NET and includes numerous controls, rich layout, and styling. It also supports vector-based graphics and animation APIs. Select Silverlight controls have been themed with a new exciting look explicitly for the Windows Phone 7 Series CTP platform. Developers can use their previous Silverlight and .NET development experience to facilitate working with this mobile control set and applying to their applications.

The following sections cover:

- ✓ **Input Types**
- ✓ **User Interface Framework**
- ✓ **Windows Phone Application Controls**
- ✓ **Windows Phone System Controls**

Input Types

Windows Phone 7 Series CTP applications have several methods available for interaction including touch gestures, an on-screen keyboard, Windows Phone hardware buttons, and sensors such as an accelerometer. This section will educate developers on how to utilize these input types in their application and provide design considerations for the development process.

The following sections provide details for various input types:

Designing for Touch

The Windows Phone 7 Series CTP user interface is designed for touch interaction, offering full navigation using a combination of finger gesture movements.

Supported Touch Gestures

Provides examples of touch gestures and design considerations.

Hardware Buttons

Provides an overview of the Windows Phone 7 Series CTP hardware button layout.

Keyboards

The on-screen keyboard is used to input text on a touch-based screen.

Sensors

Windows Phone 7 Series CTP sensors include Windows Phone accelerometer.

Designing for Touch

The Windows Phone 7 Series CTP user interface is designed for touch interaction, offering full navigation using a combination of finger gesture movements. Usability and intuitive design should be a primary goal in your application with an emphasis on well-placed and properly sized touch-based controls. The following sections will provide you with the guidance on how create such an experience for your Windows Phone application. Although this section provides touch guidance for customizing UI visual elements, we recommend that you use the standard Metro-themed controls provided in your application. These controls have been properly sized for touch interaction and based off the guidelines presented in this section.

Note:

There are exceptions to the above such as game development in which visual element touch sizing may vary from application to application.

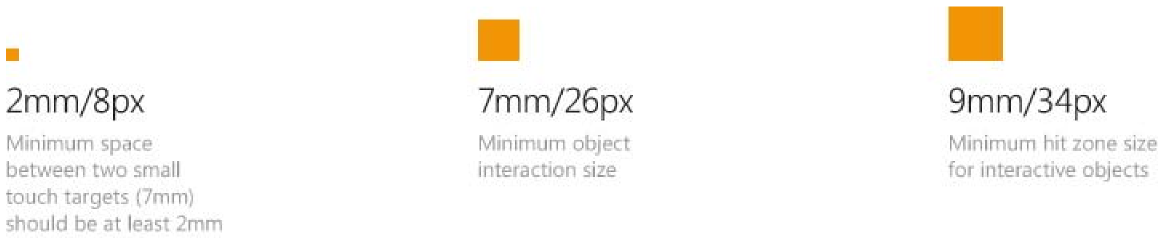
Usable and Accessible Layout

Every touchable UI element should be comfortably hit with a finger. This involves manipulating size, spacing, location, and visuals to reduce the difficulty in acquiring a target through finger touch.

Minimal Touch Target Sizes

For the following sections, if you are using pixel measurements, you should convert millimeters for the touch target size into pixels for the resolution and screen size.

- ✓ The recommended touch target size, height and width, for every UI element should be at least **9 mm**.
- ✓ The visual size, height and width, for every UI element must be between 60-100% of the touch target size.
- ✓ A touch target is not visible, so it can be larger than the visual element. However, the touch target should never be smaller than the visual element size.


2mm/8px
Minimum space between two small touch targets (7mm) should be at least 2mm
7mm/26px
Minimum object interaction size
9mm/34px
Minimum hit zone size for interactive objects

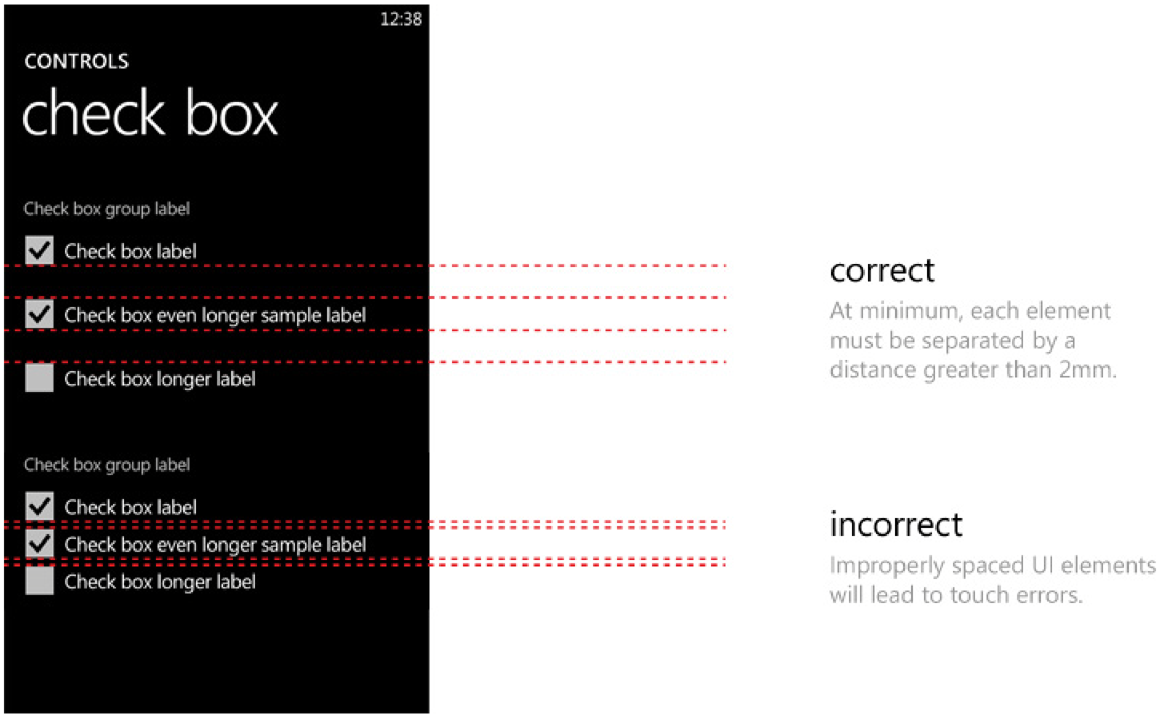
Considerations for Larger Touch Targets

The following cases require every target size to be larger than **9 mm:**

- ✓ The UI element is frequently touched.
- ✓ The result of a touch error is severe, where the user hits the wrong target. For example, when a partly typed SMS is generated and sent. Or perhaps when the error results in the deletion of an e-mail message.
- ✓ The result of the touch error is frustrating. For example, when a new dialog screen or application is launched, or when a user navigates to an undesired screen.
- ✓ The UI element is located towards the edge of the screen and is more difficult to hit.
- ✓ The UI element is being used as part of a sequential task, or a task that requires more than one hit on a series of adjacent targets on the screen. For example, when you use the dial pad.

Note:

The on-screen keyboard and hyperlinks in Windows Phone® Internet Explorer® are an exception because they have differently sized hit targets.


12:38
CONTROLS
check box
Check box group label
Check box label
Check box even longer sample label
Check box longer label
Check box group label
Check box label
Check box even longer sample label
Check box longer label
correct
At minimum, each element must be separated by a distance greater than 2mm.
incorrect
Improperly spaced UI elements will lead to touch errors.

Considerations for Smaller Touch Targets

The smallest height and width ever used for a UI element should be 7 mm.

Spacing between every two UI elements, or touch targets, must be larger than **2 mm** for targets lower than 7 mm such as a text or a radio button.

Dealing with Sizing Constraints

For UI elements with special sizing constraints, use one or more of the following mitigation methods:

- ✓ Increase spacing between UI elements so a 9 mm touch target size can be accommodated.
- ✓ If vertical size is limited, use oblong controls. These shapes are easier to hit. The touch target height can go as low as **7 mm** assuming that the touch target width is larger than **20 mm**.

Natural and Intuitive Interaction

The Metro design principles are based around the concept that UI elements should be authentically digital. You should use digital metaphors where natural and appropriate and do not necessarily try to mimic real world interaction if it is not appropriate. Developers should be aware of the concept that UI should look and feel great despite that the visuals only imitate real life behaviors. Also, if necessary, try to mimic real world behavior through direct manipulation.

Content is the Interface

- ✓ Use the application content such as a photo or a web page as the UI, and let the user interact with it using touch.
- ✓ Again, you should use the Windows Phone 7 Series CTP common Silverlight controls to maintain consistency.

Natural Response

- ✓ Use gestures to imitate real life behavior like panning on a photo to move it.
- ✓ Do not use gestures as a shortcut to a task, and only use a gesture in a manner as it was intended. For more information about touch gestures and function, see Supported Touch Gestures.

Simple and Consistent

- ✓ In Windows Phone 7 Series CTP, there are single-touch gestures that make interaction easier with one hand. There is also a multi-touch gesture available that requires a posture shift, but more advanced gesture functionality. For detailed information about the Windows Phone 7 Series CTP touch gestures, see Supported Touch Gestures.
- ✓ All basic or common tasks should be completed using a single finger.
- ✓ Response to gestures should be consistent across the phone. Using the supplied Metro-themed controls will help with maintaining consistency.

Aware and Responsive Performance

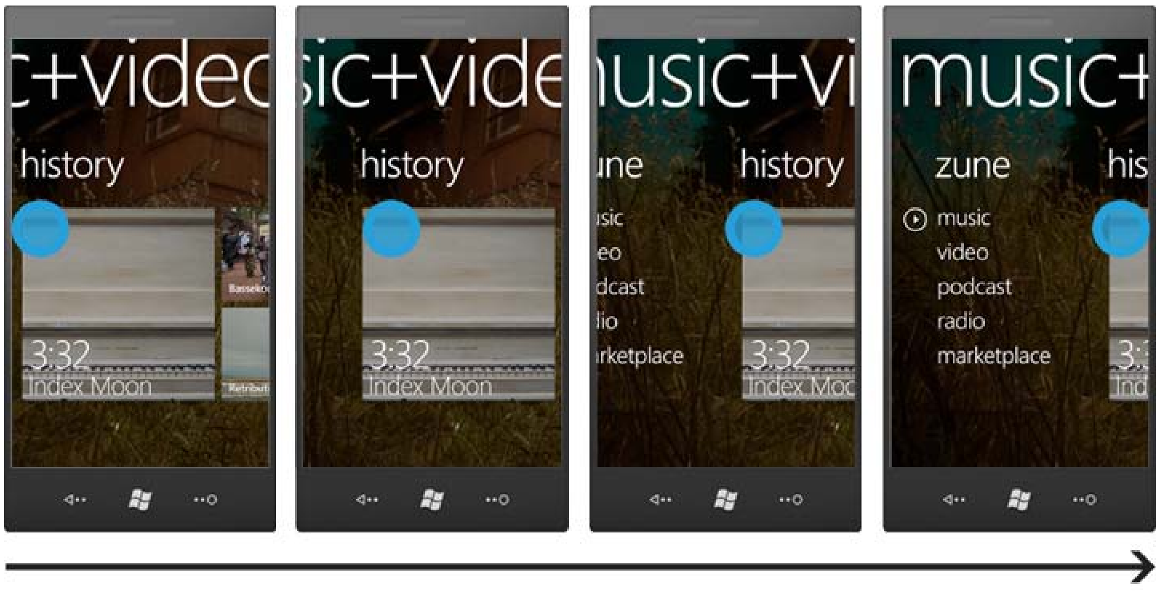
Real world objects respond to touch immediately. We strongly recommend that your Windows Phone 7 Series CTP UI abides by that rule. UI that lags or that seems slow when transitioning will have a negative impact on the user experience.

Touch Feedback

- ✓ Provide visual feedback to indicate interaction with the UI element. For example, Windows Phone 7 Series CTP uses a "tilt" animation on touch. Similarly, photos and maps do not tilt. Also, you can use motion as well to indicate interaction with an element.
- ✓ Auditory feedback is another way to indicate element interaction.

Super Responsive UI

- ✓ UI elements should respond immediately to touch. Users should not have to wait as it breaks their immersion, flow, and concentration. All actions should have immediate and obvious consequences.
- ✓ UI elements should respond while the gesture happens, not afterwards. An example would be a user flicking a photo and the movement occurs after the gesture is completed.
- ✓ Gestures can transition, or chain, from one to the other, for example, a pan may turn into a flick or a tap can become a double tap.



gesture swipe right

Your application should respond immediately to a gesture touch. For example, there should be a smooth transition from one page with no lag time. Also, the transition should occur during the movement, not at the end.

Time Consuming Processes

- ✓ If the user is faced with a time consuming process, you should elegantly provide feedback to let them know something is happening through a progress bar. Use content to indicate progress. For example, show more and more of the content as it is been downloaded. The progress bar control shows in-application activity related to an activity or a series of events. This control can be custom built by a developer; however, it is available in Windows Phone 7 Series CTP. You should use a progress bar only if needed and a message notification as a last resort. For more information, see Progress Bar.

Unique and Exciting Experience

When using a Windows Phone application, users should feel a sense of wonder while interacting with the phone. Engage your audience by promoting navigation, exploration, and exciting visuals in your design.

Discovery

- ✓ Encourage the discovery of content through using touch gestures. Users should enjoy the path as well as the completion of the task.

Please the Senses

- ✓ Touch gestures should provide a delightful experience to your application whether a single gesture is initiated, or a combination of chained gestures. Touch delights the senses as the user gets to see the interaction match the performance. Single, multi-touch, and even chained gestures can provide a more colorful, intuitive experience.

Supported Touch Gestures

A touch gesture is performing a movement with single or multiple fingers on a touch screen. Tapping a UI element such as a push button is an example. Touch gestures are the primary method for a user to interact with a Windows Phone. The following provides a detailed reference for the touch gestures supported in Windows Phone 7 Series CTP, usage and behavior. You use the provided Metro-themed controls as touch interaction elements. Again these controls have been properly sized for touch interaction. There are single and multi-touch gestures available in Windows Phone 7 Series CTP:

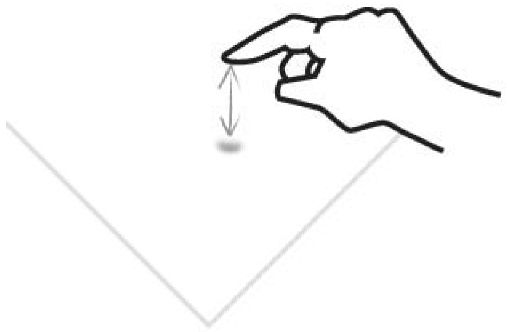
Single-touch:

- ✓ Tap
- ✓ Double Tap
- ✓ Pan
- ✓ Flick
- ✓ Touch and Hold

Multi-touch:

- ✓ Pinch and Stretch

Tap



Single touch on the screen.
Finger down on a single point within a bounded area and back up within a short period of time.

There are two behaviors associated with a tap gesture:

- ✓ Finger down provides touch indication
- ✓ Finger up executes the action

A tap also stops any type of content from moving on the screen.

Double Tap

Two quick taps within a bounded area.

The double tap toggles between **in** and **out** zoom states of a control or an application. The application will decide on its current zoomed state and will zoom in or zoom out accordingly. The zoomed-in and zoomed-out states are defined by the application.

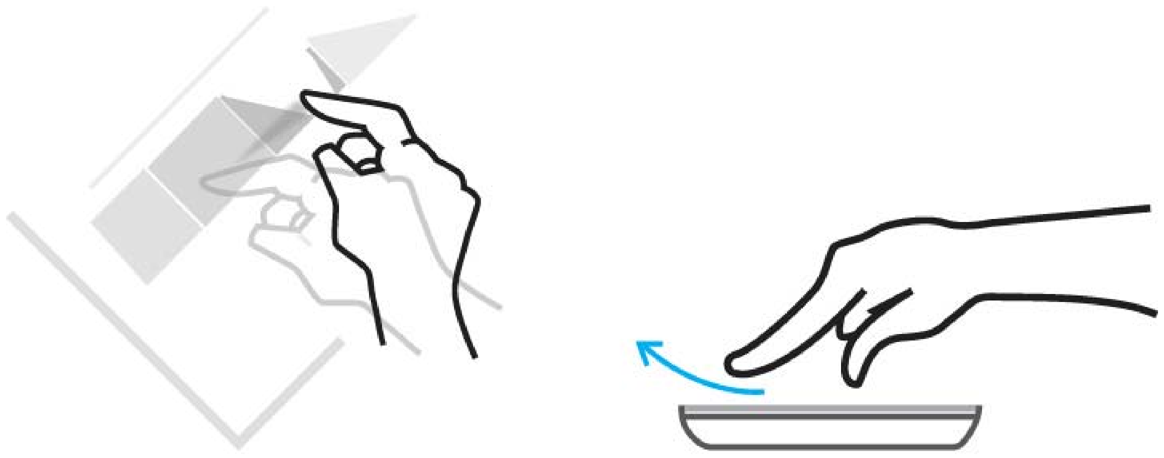
Pan

Finger down followed by finger move in a single or multiple directions. Pan ends on finger up (or when another gesture starts).

There are two behaviors associated with a pan gesture:

- ✓ Content can be moved through direct manipulation, it will stick to the finger and follow. Controls or the application can decide what panning direction to support. This can be horizontal, vertical, or any other direction specified. Also, if content is moved to an in-between state, the content should snap back to the closest state.
- ✓ A pan can move or reorder a specific item. An item follows the finger and drops in the new location when the finger is lifted.

Flick



Finger down followed by a quick finger move in a single direction and finger up. Flick can also follow a Pan gesture.

A flick gesture moves content from one area to another area. The controls or the application can decide what flicking direction to support. This can be horizontal, vertical, or another specified direction. If horizontal or vertical paths are selected, movements in other directions will be converted into vertical or horizontal movement.

Pinch and Stretch

Two fingers down followed by moving the finger closer/further apart from each other.

Pinch and stretch provides continuous zoom on the content with the center of the zoom being the center of the two fingers.

Touch and Hold

Finger down on a single point within a bounded area for a defined period of time

The touch and hold gesture shows a context menu or options page for an item.

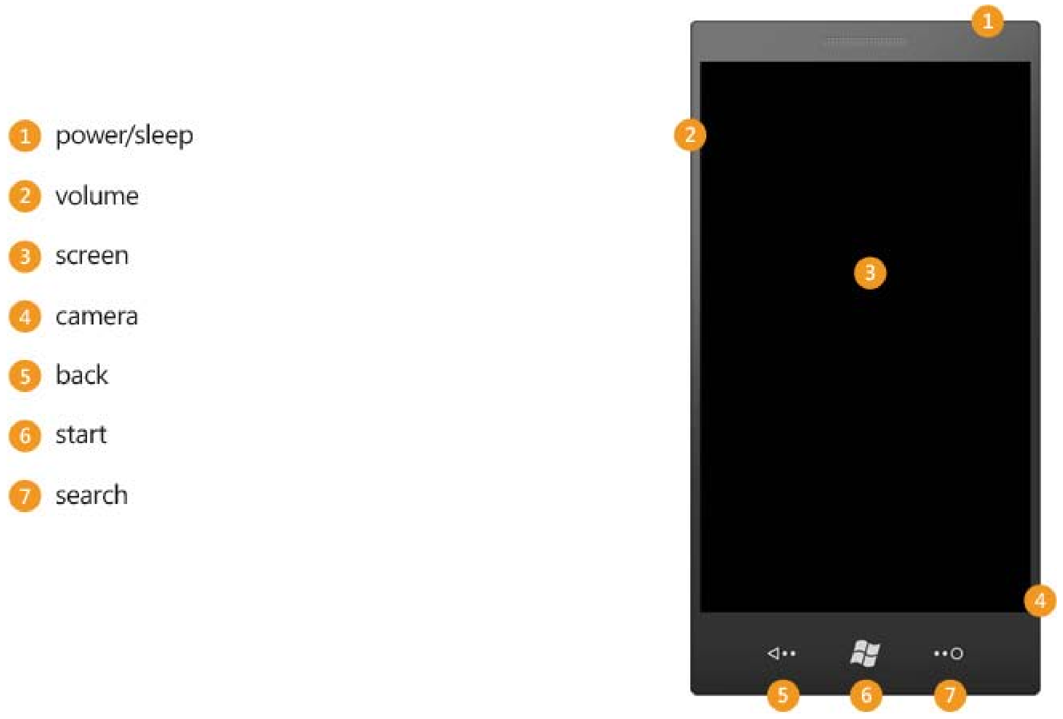
Touch Design Considerations

The following are touch design considerations for your application:

- ✓ All Windows Phone 7 Series CTP Silverlight controls will have built-in support for the touch gestures discussed in this section.
- ✓ Consistent gesture behavior must be maintained. If you create custom controls or UI, they should respond to gestures in a similar manner.
- ✓ Windows Phone 7 Series CTP gestures align with Windows desktop gestures. There is some intentional variability due to the differences in screen size and the Windows desktop support with the mouse. These differences are mostly around editing shortcuts, which can be potentially addressed by the on-screen keyboard. Applications on Windows Phone 7 Series CTP should try to align with the gesture experience of the corresponding Windows desktop application.
- ✓ Gesture extensibility is not supported in Windows Phone 7 Series CTP. You can use only the supported gestures in this topic and replicate movement as specified.

Hardware Buttons

A Windows Phone has several hardware buttons positioned around the device. Each button provides a unique function that may adjust or impact a running application.



Note:

The functionality of the hardware buttons may not be extended or overridden programmatically in your application.

Start Button

When the user presses the hardware button, it takes them to the start menu from anywhere on the device. This includes a running application.

Search Button

The search button will launch the search experience for the user to find content from anywhere on the device.

Back Button

The hardware back button is used to go back within an application or between applications. The application will allow the framework to perform the operation when the button is pressed. Also, the back button can close an on-screen keyboard, menus, dialogs, navigate to a previous page, exit a search operation, or even switch applications. However, the principal usage is to move from a current screen to the previous screen.

Note:

The back button will not work as a backspace key to delete text input.

Camera Button

From a functional standpoint, the Windows Phone 7 Series CTP camera button is dual action supporting full and half press modes. When a user initiates a full press, the phone will launch the camera application. If the user does a half press, the auto-focus feature is enabled. From a programmatic standpoint, applications will be able to launch the camera experience by calling the camera control API. Also, applications will be able to decide when and where the API is invoked.

Note:

Again, applications cannot overwrite the hardware camera button behavior.

Volume Buttons

The volume buttons are primarily used to adjust the volume of an active audio stream. This could be a phone call, music audio, radio audio, or an in-progress movie. When the audio stream is not active, the volume buttons are used to change the active sound profile. The volume control also exposes the audio transport controls such as previous and next.

For the volume buttons:

- ✓ If there is an incoming call, the volume buttons will silence the ringtone.
- ✓ For an active audio stream playing, the volume buttons will change the volume level for the active audio stream. System sound events and effects are exempt.
- ✓ When a device is locked, the volume buttons should still be active and respond to the action taken. A developer cannot override this functionality.
- ✓ When a user presses and holds on a volume button, it will do a key press repeat. Holding the volume up key will incrementally increase the volume.

Power Button

If the phone is in use, a brief press of the power button will turn off the display and lock the phone. When the phone is not in use, a brief press of the power button will wake up the phone.

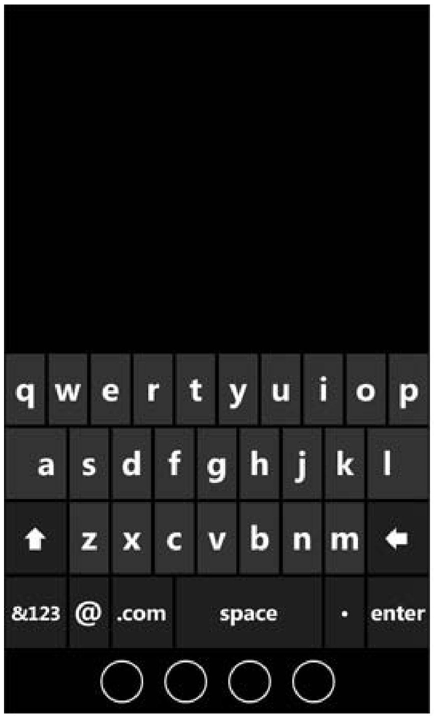
Keyboards

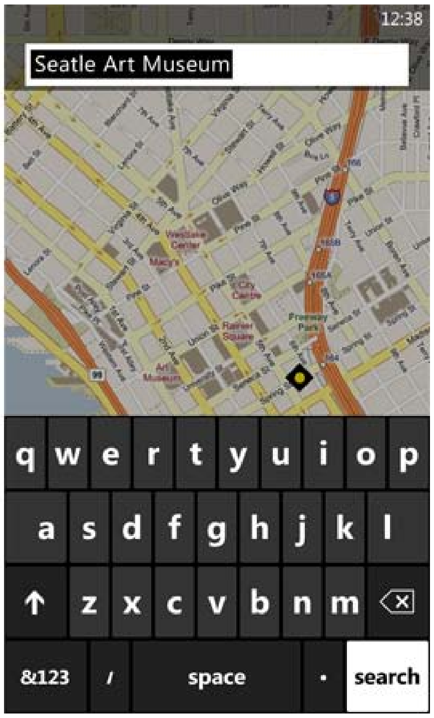
On-Screen Keyboard

The on-screen or hardware keyboard can be used to input text on a Windows Phone. The on-screen keyboard is deployed automatically when an edit control becomes active. It closes when the user taps outside of the edit control. If a phone includes a hardware keyboard, the on-screen keyboard automatically closes when it is deployed. The Windows Phone 7 Series CTP phone features several typing aides such as text suggestions and auto-correction. These features apply to both the on-screen and hardware keyboard.

Note:

Whether a hardware keyboard is utilized on the phone will depend on the manufacturer.





On-Screen Keyboard Design Considerations

- ✓ When the keyboard is deployed in an application, it covers the bottom area of the screen.
- ✓ If text suggestions are enabled they are docked above the keyboard and they take additional space away from the application.
- ✓ When the keyboard is deployed, the application should scroll to ensure the active edit control and the caret are in view. If using a multi-line edit control, part of it may be hidden behind the on-screen keyboard. You must ensure that the line containing the caret is always in view and above the on-screen keyboard.
- ✓ In some applications it might be difficult for the user to tap outside of the text box or edit control to close the on-screen keyboard and view more content. An example, when an application takes up all or most of the screen area. The application can choose to automatically close the on-screen keyboard when the user tries to view the content rather than type. For example the user scrolls the application area or the active edit control gets outside of the viewable area. Another option is to implement a read view and an edit view. Then the user switches to the edit view, the application can deploy the on-screen keyboard. When the user switches back to the read view, for example by pressing the back button, the application can choose to close the on-screen keyboard.
- ✓ The keyboard supports multiple key layouts depending on the text box or edit field context.
- ✓ Developers should set an input scope in edit fields to choose the appropriate layout and enable the appropriate typing aides. For example, if a developer chooses the URL input scope, a keyboard layout will be shown featuring a .com key. To accomplish this you need set the **input scope** property in your project for the text box or edit control.

There are several context specific keyboard layouts such as the following:

Keyboard Type	**Layout**
Default	Standard QWERTY layout
Text	Standard layout with ASCII based emoticons
Email Address	Standard layout with .com and @ keys
Phone Number	Typical 12-key layout
Web Address	Standard layout with .com key and customized Enter key
Maps	Standard layout with a customized Enter key
Search	Semi-transparent layout with a Search and .com key
SMS Address	Standard layout with easy access to phone number layout

Hardware Keyboard

Having a hardware keyboard on your Windows Phone will be dependent on the manufacturer of the phone. There are several hardware keyboards that may come available in the Windows Phone 7 Series CTP including such designs as a pull out bar, a vertical slide configuration, or even a flip or swivel orientation.

Note:

Windows Phone 7 Series CTP supports only full alphabet layouts such as QWERTY, AZERTY, and QWERTZ. 12/-20 key layouts are not supported.

The hardware keyboard allows for typing letter, accented letters, numbers and symbols:

- ✓ The shift key allows for typing capital letters. There are 3 shift modes: On, Off, and Lock (Caps Lock).
- ✓ The emoticon key brings up the emoticons picker.
- ✓ The accent key is used to type accented letters. When you press the accent key it adds an accent the letter left of the caret. Multiple presses cycle through the available accents. The function key (FN) plus the accent cycles back to the previous accent. If you press and hold on the accent key it brings up an accent picker.
- ✓ Characters that are not available on the keyboard are accessible via a symbol picker that is launched by pressing the symbol key (SYM).
- ✓ If you press and hold on the SYM key, it brings up the language picker. FN and SYM switches to the next language.

- ✓ A system tray input indicator represents shift mode, FN mode, and active language.
- ✓ Keyboard keys can be overloaded. You can press and hold on a key or using the FN key allows for accessing secondary functionality.

Hardware Keyboard Design Considerations

- ✓ The hardware keyboard is only used for typing and not for controlling the UI.
- ✓ The hardware keyboard can include arrow keys that can move the caret within an edit control. However these arrow keys must not be used to move focus, scroll lists, navigate maps or web pages.
- ✓ The following keys will always be available on the hardware keyboard:
  - ✓ Letters (A-Z), Enter, Space, Backspace, Shift, emoticon, SYM, period and comma.
  - ✓ Numbers (0-9) as either a primary or secondary character.
  - ✓ Accent key for German, French, Italian and Spanish keyboards.
- ✓ The following keys are no longer supported on the hardware keyboard:
  - ✓ The directional pad or any other navigation specific hardware.
  - ✓ The "OK & Home" and the "Send & End" hardware soft keys.
  - ✓ The keys Delete, Insert, Control (CTRL), Alt, Caps Lock, Tab, page up and down, and escape (ESC).
  - ✓ The start, search, and back keys as part of the keyboard.
- ✓ When the symbol picker, accent picker or language picker are launched, they are located at the bottom part of the screen. They go away after a selection is made or a predefined timeout.
- ✓ Applications can use an API to know when the hardware keyboard is available or deployed and act accordingly.
- ✓ When the hardware keyboard is deployed, the on-screen keyboard will close. If the device has a fixed hardware keyboard, the on-screen keyboard will never be deployed.
- ✓ The typing aides are also available for the hardware keyboard.

Sensors

Accelerometer

The Windows Phone 7 Series CTP accelerometer is an electromechanical device that measures acceleration caused by gravity or external sources. A 3D motion sensor that provides continuous information about the forces being applied to the device in the X, Y, and Z planes. A Windows Phone can leverage this feature to create sophisticated experiences for the end user. Some goals of the design are to provide managed APIs that are easy to use and flexible to offer scenarios such as automatic screen rotation, tilt-to-scroll, and gaming.

Application Design Considerations

Developers creating applications or games that require a higher level of precision and sensitivity need to have the data calibrated.

Note:

For the CTP release of the Windows Phone 7 Series CTP application platform, there is no exposed API for calibration. This functionality will be added at a later release.

User Interface Framework

The Windows Phone 7 Series CTP user interface is modeled to provide an optimal viewing experience for the end user. This section will educate developers on topics such as full screen mode, screen orientation, frame and page navigation, and application scrolling.

The following topics are discussed in this section:

Screen Orientation

Supports two types of orientation, portrait and landscape.

Frame and Page Construction

Windows Phone 7 Series CTP applications are based on a Silverlight page model where users can navigate forward through different screens of content.

Scroll Viewer

Application scrolling occurs when content exceeds the screen size.

Themes

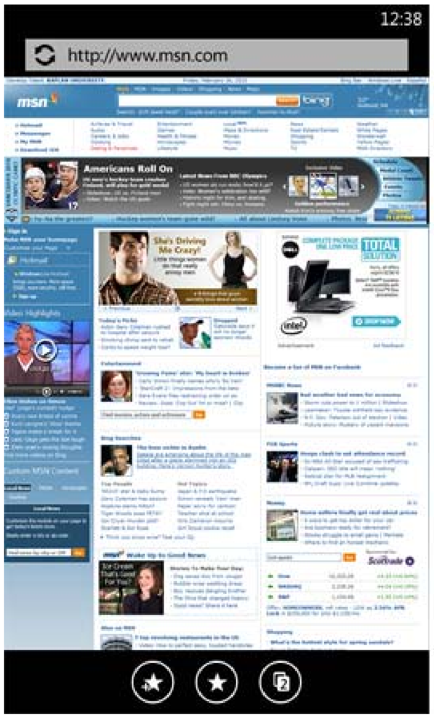
A theme is a resource used to personalize the visual elements on a Windows Phone.

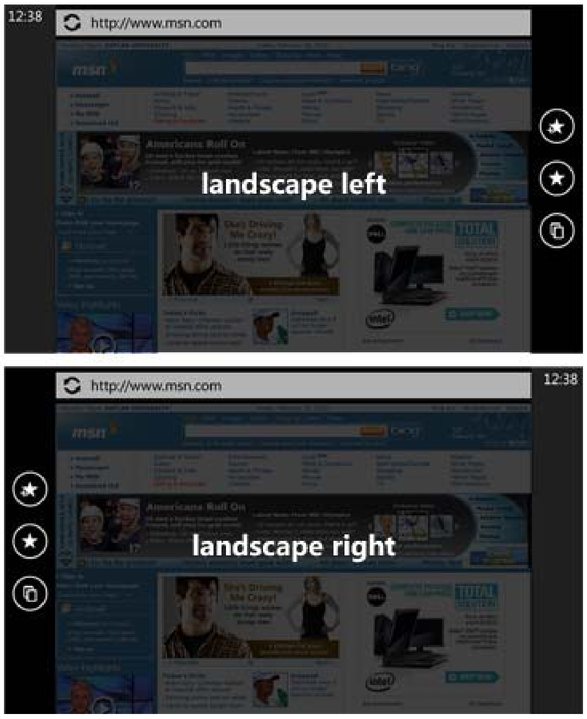
Screen Orientation

Windows Phone supports three types of orientation: portrait, landscape left, and landscape right. The difference between landscape left and landscape right is the orientation of the application bar and system tray.

Note:

Applications cannot specify only left or only right configurations for application bar and system tray placement.





Portrait Orientation

In portrait-mode, the page is vertically oriented where the height of the page is greater than the width. The orientation can change based upon events such as device rotation, or sliding out a hardware keyboard. When the orientation is switched applications that are written to be orientation-aware will respect the new orientation.

Landscape Orientation

In landscape-mode the device is moved to its side where the width of the page is greater than the height. The following offers some uses for landscape orientation:

- ✓ Use for any type related input such as email or text messaging using the slide out landscape keyboard if available.
- ✓ It provides a better browsing for a webpage that is suited for landscape mode.
- ✓ Taking a picture with a camera
- ✓ Viewing a single photo or watching a slideshow
- ✓ Watching a movie or video
- ✓ Playing games

Application Design Considerations

The following are design considerations for using portrait and landscape mode in your application.

- ✓ Applications must be configured to support multiple orientations. If a specified page on the application supports both landscape and portrait orientation, the user can initiate a change by physically rotating the device or opening up a portrait or landscape keyboard. There is no programmatic way to directly switch orientation as the orientation property is set to read-only. The indirect method to accomplish this is to set the Supported Orientations property to the desired orientation.
- ✓ When changing to landscape mode, the following will become landscape-aware in the application: the system tray and application bar, the application menu, volume user interface component, notifications, and system dialogs such as low battery notifications.
- ✓ Text input in portrait or landscape mode will either be through a hardware keyboard, when available, or the on-screen keyboard.

Frame and Page Construction

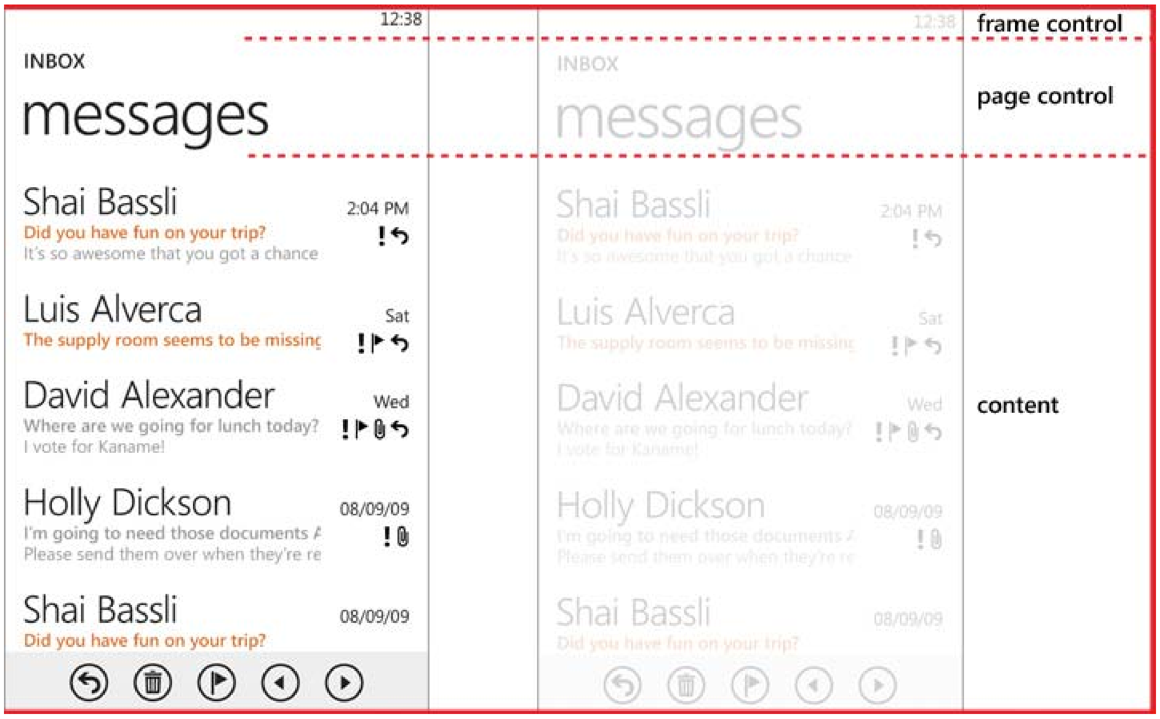
This section will detail frame and page construction for your application. It will also cover the custom UI framework design for Windows Phone 7 Series CTP.

Note:

This section will be expanded upon in future releases of this document.

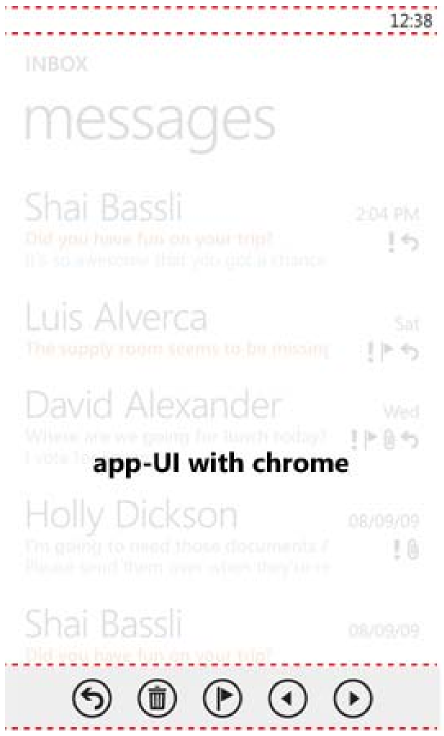
Windows Phone 7 Series CTP applications are based on a Silverlight page model where users can navigate forward through different screens of content. Also, you can move backward using the Windows Phone "back" hardware button. A goal of this model is to ease the creation of view-based applications that fit naturally into the Windows Phone 7 Series CTP navigation model.

The core elements include a top-level container control called a PhoneApplicationFrame (frame) that can host a PhoneApplicationPage (page). Pages hold discrete sections of content in your application. Windows Phone 7 Series CTP provides frame and page classes to facilitate navigation to separate sections of content. You create as many different pages as needed to present the content in your application and then navigate to those pages from the frame. The following illustrates a possible frame and page hierarchy for an application:


12:38
INBOX
messages
Shai Bassli
Did you have fun on your trip?
It's so awesome that you got a chance
2:04 PM
Luis Alverca
The supply room seems to be missing
Sat
David Alexander
Where are we going for lunch today?
I vote for Kaname!
Wed
Holly Dickson
I'm going to need those documents
Please send them over when they're re
08/09/09
Shai Bassli
Did you have fun on your trip?
08/09/09
frame control
page control
content

Custom Screen UI Interface

This is a state where the page switches context from normal-view to full-screen view. The application can choose on whether to include the system tray or application bar at this time. However the developer should implement the behavior as it is not automatic. An example is a game is started on the Windows Phone and the game settings are opened. From that settings menu, the user may start a new game and the page would be launched into full-screen mode.



12:38

INBOX

messages

Shai Bassli 2:04 PM
Did you have fun on your trip?
It's so awesome that you got a chance

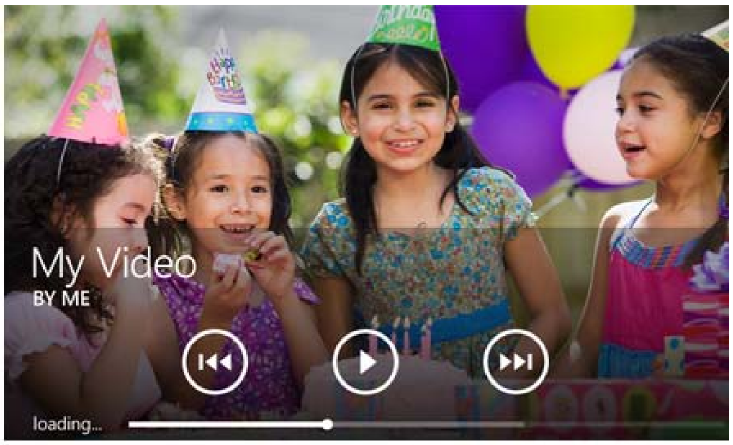
Luis Alverca Sat
The supply room seems to be missing

David Alexander Wed
Where are we going for lunch today?
I vote fo

Holly Dickson 08/09/09
I'm going to need those documents f
Please send them over when they're re

Shai Bassli 08/09/09
Did you have fun on your trip?

app-UI with chrome



app-UI without chrome

Note:

The above is considered a best practice for using custom-full UI mode. While the application is in full-screen only, all visual notifications including incoming calls will be allowed to be passed to the UI as normal.

When your application launches into full-screen mode, the system tray and application bar are not visible. The developer can hide or surface the application bar in their application. There is a visibility property (Page.FullScreen) that allows you to either show or not show the application bar and system tray.

Scroll Viewer

Application scrolling occurs when content in a scroll view control exceeds the bounds of the scroll view control. The contents are wrapped in a scroll panel that would handle the page scrolling. This will also enable the scroll indicator to indicate whether the content is longer or wider than the page, and the current position on the page.

Application Design Considerations

- ✓ The scroll indicator is positioned on the right side for vertical scrolling and along the bottom for horizontal scrolling. The scroll indicator will only appear when you interact with the page.
- ✓ The scroll indicator times out after a short time when there are no gesture movements applied to the screen such as flicking and panning. It comes back into view when these gesture touches are made.
- ✓ The scroll indicator indicates where the user is within the content. For example, if the user is at the top of the page, the scroll indicator appears at the top.
- ✓ Avoid putting list boxes inside a scroll viewer if they have a lot of content. This may degrade performance.

Themes

A theme is a resource used to personalize the visual elements on a Windows Phone. Windows Phone 7 Series CTP developers can create applications that preserve the look and feel of the native device user interface (UI) from a stylistic standpoint. These style properties include of background colors and accent colors. The theme ensures that controls and UI elements appear consistently across the platform preventing a jarring, unsettled user experience.

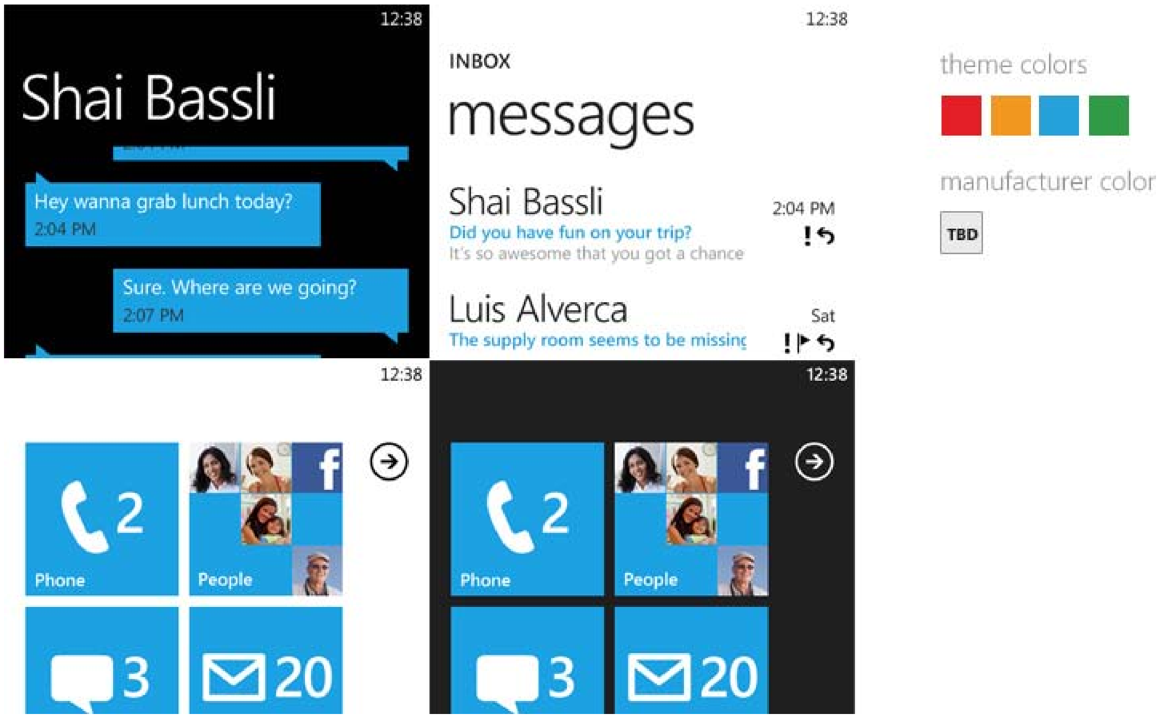
Some goals of this feature include:

- ✓ The Windows Phone 7 Series CTP developers can build applications and access the themed properties directly in their code.
- ✓ Developers can explicitly change the value of any themed property to match their own branding requirements.

An advantage of using themes in the application platform is consistency and compatibility with the Metro design principles. A developer can use the default control set without adjusting common properties such as colors knowing that these styles will be modified at runtime via the system theme file. When an application is run on a Windows Phone, it automatically takes this system theme file and modifies the visuals of the application accordingly. Also, a developer can override the theme at the application level. For example, a company is building an application with a strong brand color and wishes to maintain this color. The developer can provide their own resources and override any themed properties. However, they cannot turn off theming.

Note:

Only colors are part of a theme. Other elements such as fonts or control sizing do not change.



Shai Bassli

Hey wanna grab lunch today?
2:04 PM

Sure. Where are we going?
2:07 PM

INBOX

messages

Shai Bassli 2:04 PM
Did you have fun on your trip?
It's so awesome that you got a chance

Luis Alverca Sat
The supply room seems to be missing

Application Design Considerations

- ✓ Users can choose a light or dark theme and developers should take this into account when developing their UI. For example, if you choose an all white background throughout your application, this may have an impact on battery life on organic LED displays. During development, you should be always be aware of what application backgrounds can do to battery life.
- ✓ The user can choose between 5 different accent colors for your application. If you choose to use an accent color theme property in your application, these colors will change based on user preference. The accent colors include orange, blue, green, and red. The manufacturer of the phone has the ability to supply the last accent color.
- ✓ You cannot modify system-wide themes, you can only modify themes in your application.
- ✓ If you explicitly set the foreground or background color of a control you should verify that the content is visible in both dark and light themes. If the color you have chosen is not visible, either choose a more appropriate color or also explicitly set the background or foreground to maintain contrast.

Windows Phone Application Controls

The Windows Phone 7 Series CTP UI is based off a design system called Metro. In Windows Phone 7 Series CTP there are a collection of such Metro-themed Silverlight controls to use in your application. Metro interfaces are supposed to embody harmonious, functional, and attractive visual elements. We highly recommend that developers try to adopt the Metro design style whenever possible. Although UI requirements may vary based on the application, paralleling this experience will create a more consistent, fluid UI experience for your application. The base Metro-themed Silverlight controls for Windows Phone 7 Series CTP are listed in this topic, and also controls that are not supported in the platform. In the topics that follow this section, there will be detailed overview and guidance for a selection of the available controls.

Note:

The topics in the following section only address a selection of the Windows Phone 7 Series CTP controls set.

Supported Controls in Windows Phone 7 Series

Border	Button	Canvas	Check Box	Content Control
Content Presenter	Control	Grid	Hyperlink Button	Image
In Presenter	List Box	Media Element	Multi Scale Image	Panel
Password Box	Progress Bar	Radio Button	Scroll Viewer	Slider
Stack Panel	Text Block	Text Box	Phone Application Page	List View Item
List View	Toggle Control Switch	Phone Application Frame	User Control	

Note:

For the CTP release you must use the List Box control rather than the List View control.

Unsupported Controls in Windows Phone 7 Series

Combo Box	Message Box	Open File Dialog	Save File Dialog	Tool Tip
Calendar	Data Grid	Date Picker	Frame (Use Phone Application Frame Class)	Page (Use Phone Application Page Class)
Gide Splitter	Label	Scroll Bar	Tab Control	Tree View

For the above unsupported controls please note the following for the Windows Phone 7 Series CTP release:

- ✓ The combo box, scroll viewer, and tool tip are hidden from the toolbox but can still be used in your application. The combo box is not themed to the Metro design and you will have to do so to maintain UI consistency. The scroll bar and tool tip are not suitable controls to use in a touch-based system.
- ✓ For the CTP release there will be a message box control available. However for future releases this control may or may not be supported.
- ✓ The open and save file dialogs are not supported. You will have to use isolated storage in your application.
- ✓ The calendar, data grid, date picker, grid splitter, tab control, and tree view controls are from the Silverlight 3.0 platform and cannot be used. You will have to create your own version of these controls and theme accordingly to the Metro design.

The following topics define a selection of Windows Phone 7 Series CTP controls and provide design guidance on how to use them in your application.

Note:

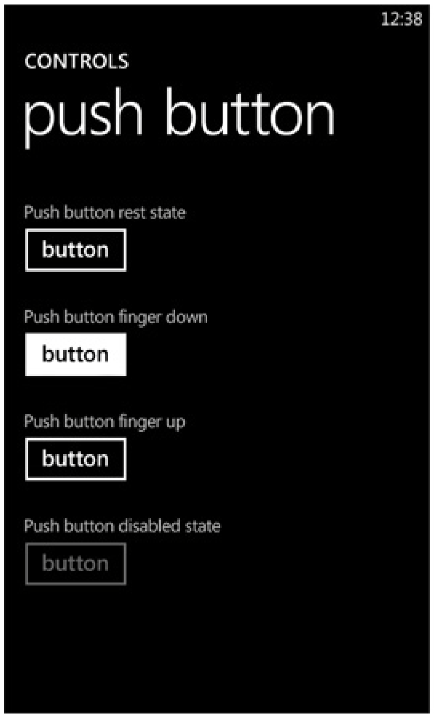
For the CTP release there are a few controls that will have to be manually styled to the Metro design: Toggle Switch, List Box, List View Item, and Text Block. For more information, see the **Base Controls** topic in the Windows Phone Development documentation.

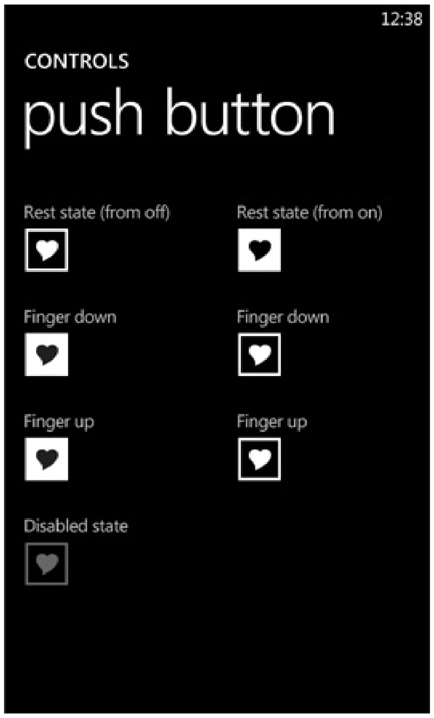
- ✓ Push Button
- ✓ Toggle Switch
- ✓ Check Box
- ✓ Radio Button
- ✓ Hyperlink Control
- ✓ Slider Control
- ✓ Text Block
- ✓ Text Box
- ✓ List Box
- ✓ List View Item
- ✓ Progress Bar
- ✓ Page Title

The following two controls are not available in the CTP release. You will have to create such experiences in your application. However these topics will provide some design guidance on how to approach this type of UI development.

- ✓ Panorama Application
- ✓ Pivot Control

Push Button





A push button initiates an action when a user taps it. The shape is usually rectangular and the standard layout allows for either text or an image.

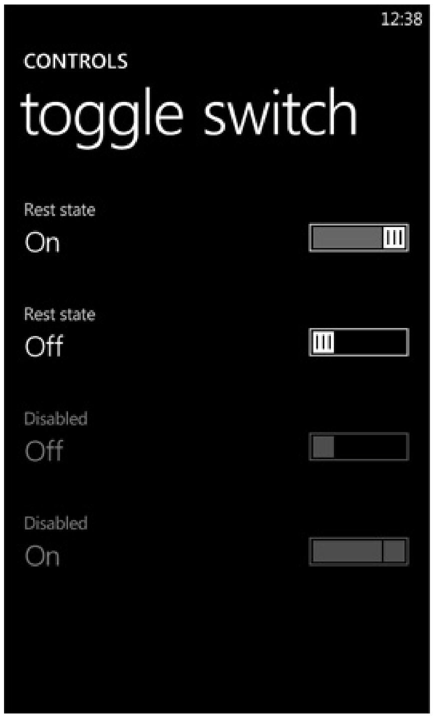
Some features include:

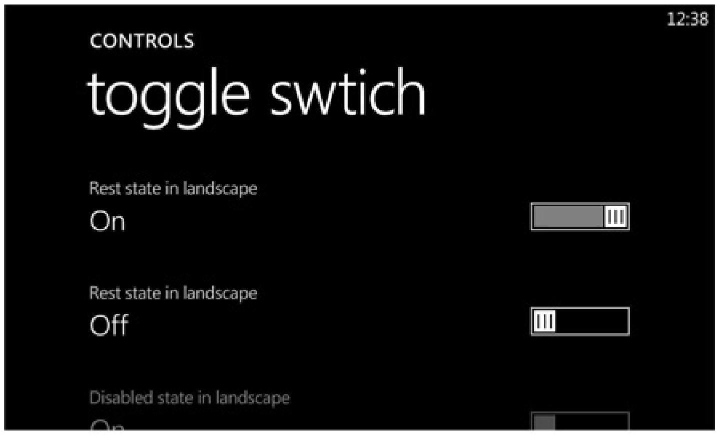
- ✓ The push button supports rest, press, and disabled states.
- ✓ There is no visible focus state.

Application Design Considerations

- ✓ Gesture support: The push button supports the tap gesture.
- ✓ The push button supports custom styling and theming.
- ✓ The push button should never include more than two words.
- ✓ The text should be concise and typically a verb.

Toggle Switch





A toggle control has one of two possible choices selected at any time, and provides visual awareness of its current state. A user can tap on the control to toggle between the two choices. For Windows Phone 7 Series CTP, the toggle switch is available to developers.

Important:

For the CTP release this control must be styled correctly when placed in your application. For more information, see the **Base Controls** topic in the Windows Phone Development documentation.

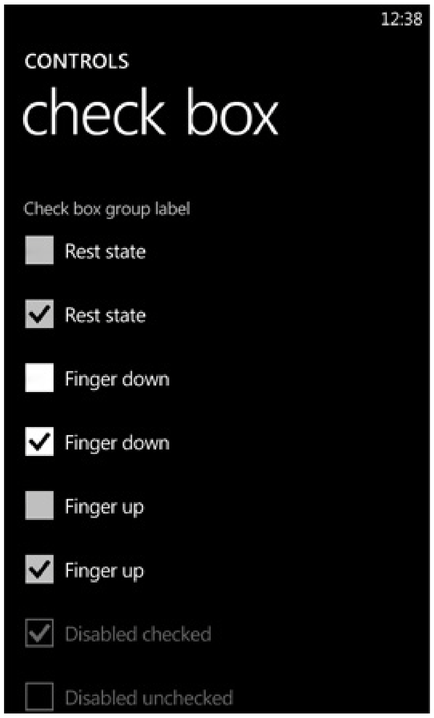
Some features include:

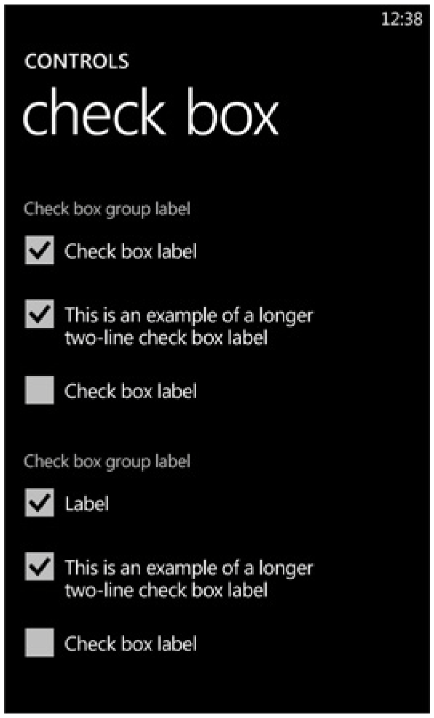
- ✓ The toggle switch supports rest and disabled states for both on and off settings.
- ✓ The toggle switch supports theming.
- ✓ There is no visible focus state on the control.

Application Design Considerations

- ✓ Gesture Support for toggle switch: The toggle switch supports the tap gesture for toggling the control.
- ✓ Toggle switches should be used sparingly.

Check Box





A check box control is used for selecting an item from a list. Check boxes can be used in groups to display multiple choices from which the user can select one or more choices. A user can either tap the box or the accompanying text to select an option. Also, although the control can support multiple lines, text should be either in either a one or two line format from a design perspective.

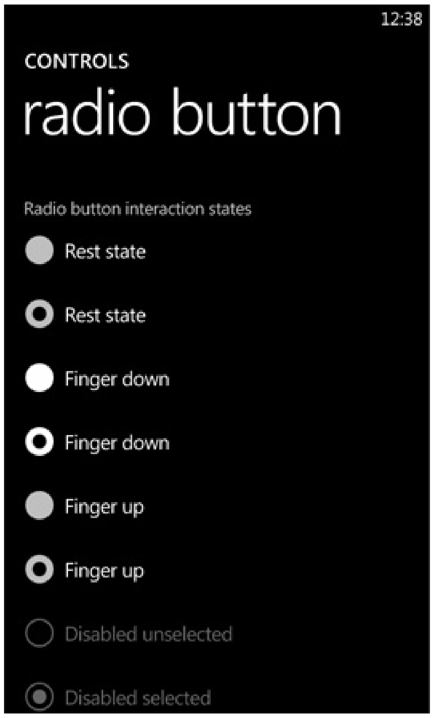
Some features include:

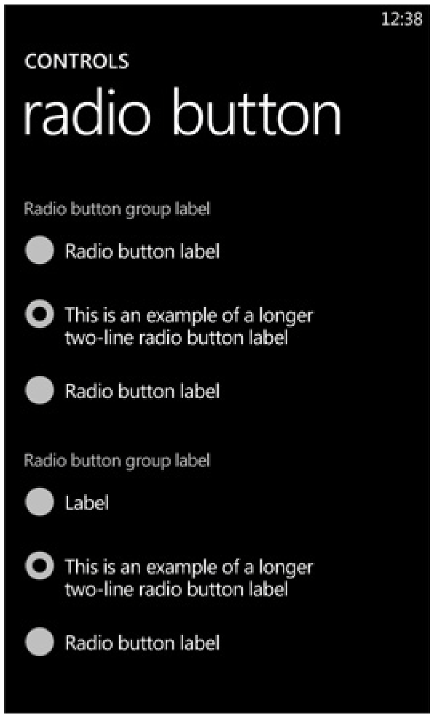
- ✓ The check box control supports rest, press, and disabled states for both selected and un-selected settings.
- ✓ There is no visible focus state.
- ✓ The check box control supports theming.
- ✓ The control also includes an indeterminate state.

Application Design Considerations

- ✓ With several choices you may consider using a scroll viewer and adding a stack panel.
- ✓ Gesture support: The check box supports a tap gesture on the glyph as well as on the label as a single hit target. This would switch between selected and un-selected settings.
- ✓ The control also includes an indeterminate state that can be used to communicate checked and unchecked status simultaneously for a number of underlying items. We do not recommend you using this state as the user may be confused as to which of the underlying property items is actually checked or unchecked. A more appropriate alternative to map the data sources for that checkbox into separate checkboxes or consider using a multi-selection list, with the latter being a potentially more attractive option for a dynamic data set.
- ✓ You should not use the checkbox control if you are looking for richer layout support including labeling.

Radio Button





A radio button is used to represent a set of related, but mutually exclusive choices. The user taps on the Radio Button text to select the control. Only one option may be selected at a time.

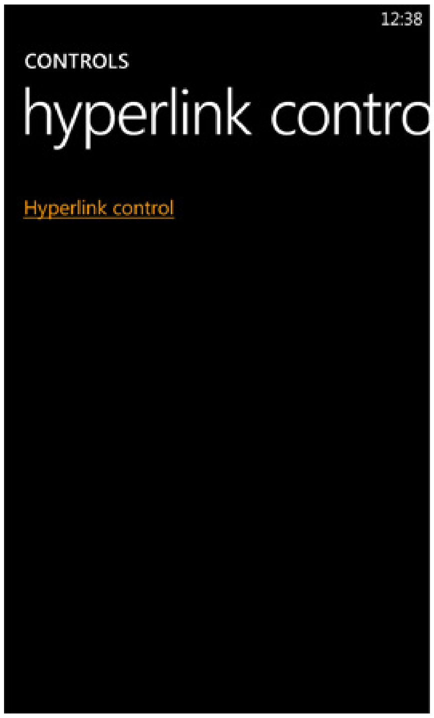
Some features include:

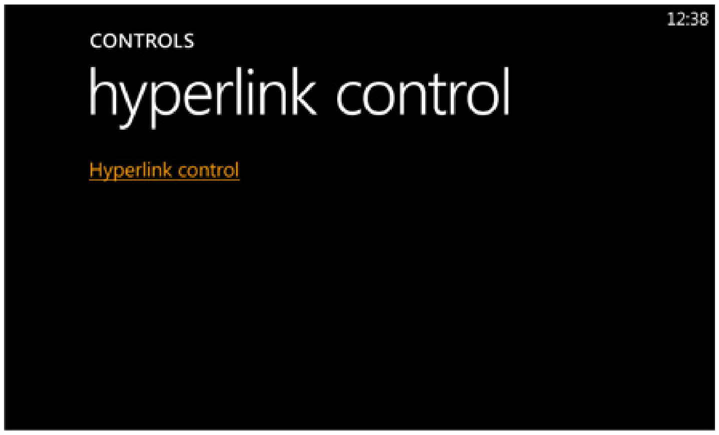
- ✓ The radio button control implements rest, press and disabled states for both selected and un-selected settings.
- ✓ The radio button control supports theming.
- ✓ There is no visible focus state.

Application Design Considerations

- ✓ You may consider using a list box with list view items if the there are several choices.
- ✓ The radio button text may wrap to a second line as the control supports multiple line formats. However from a design perspective they should use either a single or double line format. Also, applications should provide enough space for additional text.
- ✓ The control also includes an indeterminate state that can be used to communicate checked and unchecked status simultaneously for a number of underlying items. We do not recommend you using this state as the user may be confused as to which of the underlying property items is actually checked or unchecked. A more appropriate alternative to map the data sources for that checkbox into separate checkboxes or consider using a multi-selection list, with the latter being a potentially more attractive option for a dynamic data set.
- ✓ Gesture support: The radio button supports a tap gesture on the glyph as well as on the label as a single hit target. This would switch between selected and un-selected settings.

Hyperlink Control





A hyperlink control allows you to embed hypertext links in a page, and should only be used for navigation.

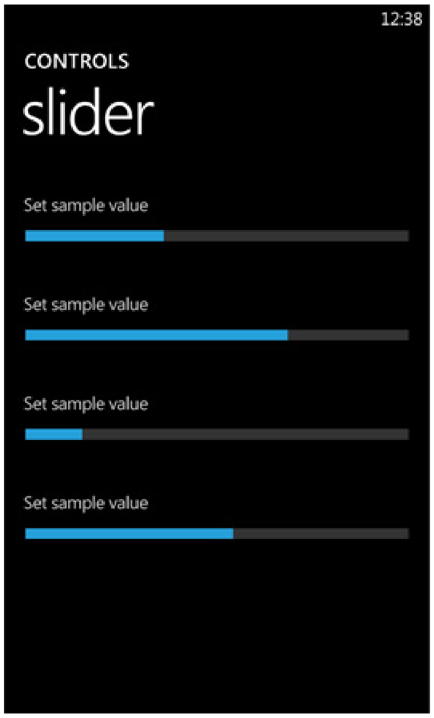
Some features include:

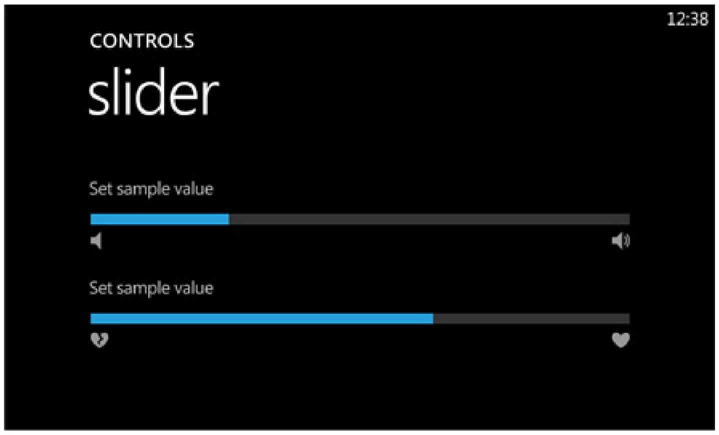
- ✓ The control supports rest and press states
- ✓ You can specify a navigation target.
- ✓ There is no visible focus state.
- ✓ There is support for theming.

Application Design Considerations

- ✓ Avoid placing many link controls in a close proximity. Doing so may make it difficult for the user to select an individual link.
- ✓ Applications should use a push button control instead of a link control for situations where a user can hide or show additional text. For example, when you use text such as "More Details..." or "More Info".
- ✓ Link controls should only use a disabled state if the state is temporary such as other system processes are occurring, or if the state can be changed to enabled by a user action. A link that is not available and cannot be enabled by user action should not be shown.

Slider Control





A slider control is used for setting a value from a continuous range of data. For example, if you set volume or brightness levels. The slider has a minimum and a maximum increment value. The control can optionally show a progress indicator when the slider track is pressed.

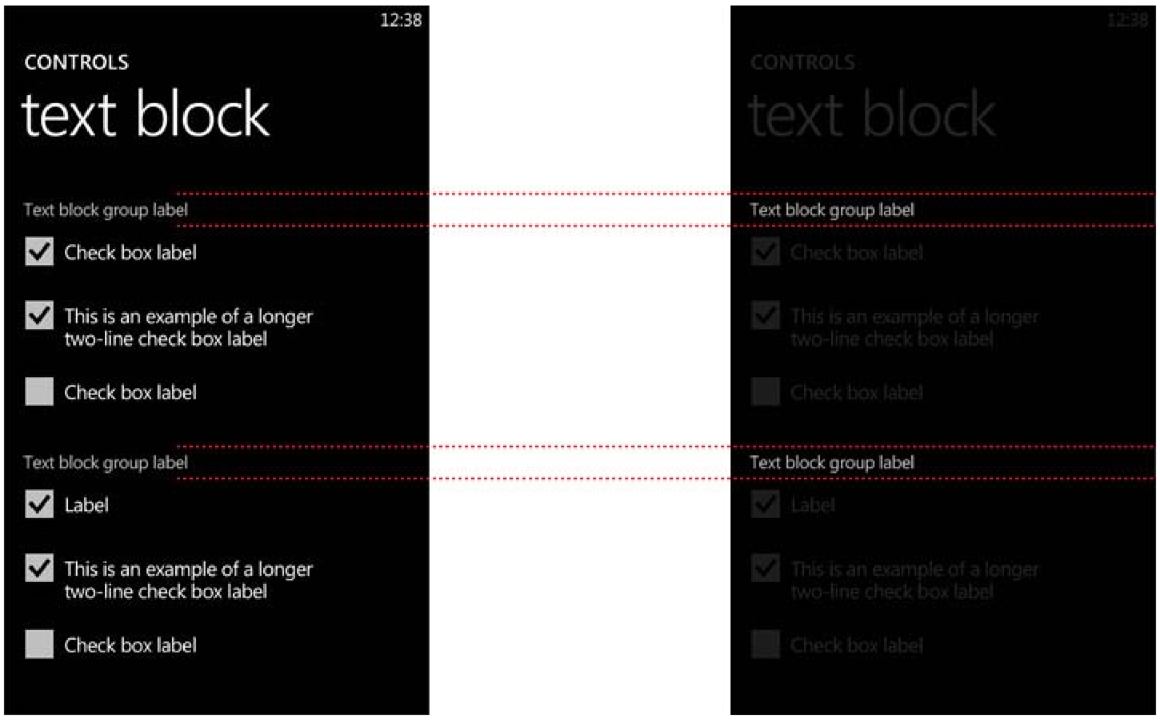
Note:

For the CTP release this control must be styled correctly when placed in your application. For more information, see the **Base Controls** topic in the Windows Phone Development documentation.

Application Design Considerations

✓ Applications can use the slider control in either a horizontal or vertical placement. However, the horizontal layout is recommended.

Text Block



The text block displays a fixed amount of text and is used to label controls or control groups. The text block stays the same for all states of the related control. This control supports theming and word wrapping.

Note:

For the CTP release this control must be styled correctly when placed in your application. For more information, see the **Base Controls** topic in the Windows Phone Development documentation.

Text Box

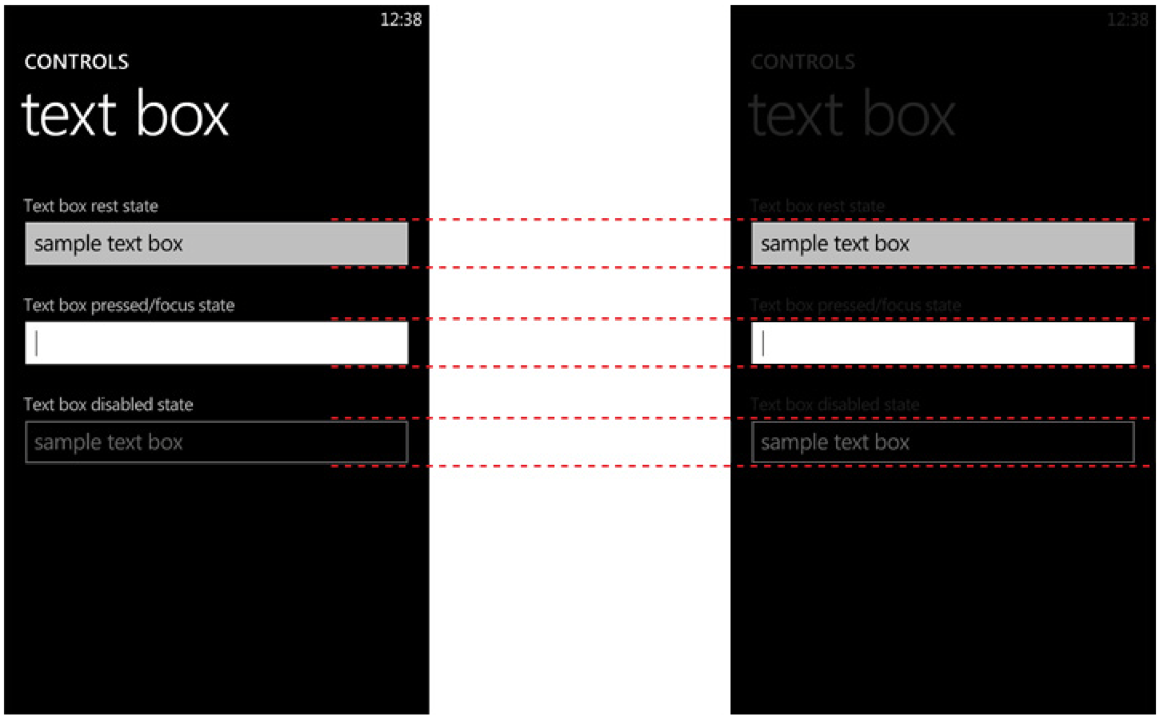
The text box control displays content, allows the user type content, or edit the contents. It may be set to read only, but is generally used for editable text. Text boxes may display a single or multiple lines. The latter causing it to wrap text to the size of the control.

Some text box features include:

✓ The text box control supports editable and read only states.

✓ Theming is supported for the text box control.

✓ The on-screen keyboard comes up automatically when focus is set in the text box.



Application Design Considerations

✓ To add an input scope for an on-screen keyboard you need to configure the input scope property on the text box control.

✓ Gesture support: Supports tap gesture for focus and selection.

✓ To get password obfuscation use a password box control.

List Box

A list box control contains a collection of items and you can populate this control by binding to a data source or displaying unbound items. The list box is an items control that you can also populate with other controls or text. For example, Windows Phone 7 Series CTP list view item controls can be placed inside with different presentation and hit target elements.

Note:

For the CTP release this control must be styled correctly when placed in your application. For more information, see the **Base Controls** topic in the Windows Phone Development documentation.

List View Item

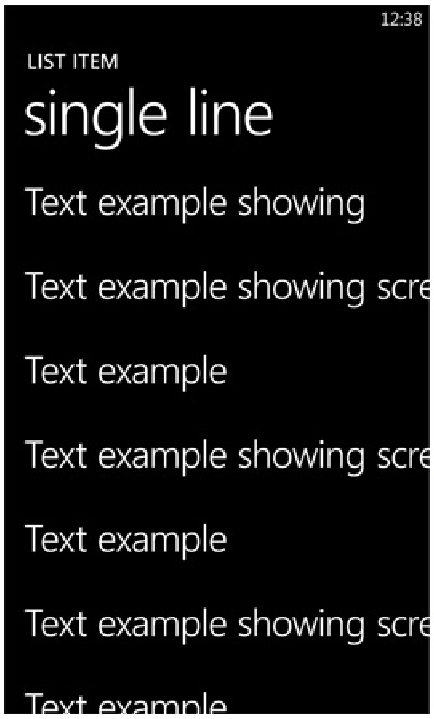
In Windows Phone 7 Series CTP, list controls provide developers a way to present data contents in an organized fashion. A list view item is a rectangular based visual element that appears in a list. For Windows Phone 7 Series CTP, there is an assortment of list view item formats available for use.

Note:

For the CTP release this control must be styled correctly when placed in your application. For more information, see the **Base Controls** topic in the Windows Phone Development documentation.

- ✓ Single Line
- ✓ Double Line
- ✓ Single Line with Icon
- ✓ Double Line with Icon
- ✓ Single or Double Line with Secondary Hit Target

One example is the single line presentation.



12:38

LIST ITEM

single line

Text example showing

Text example showing scre

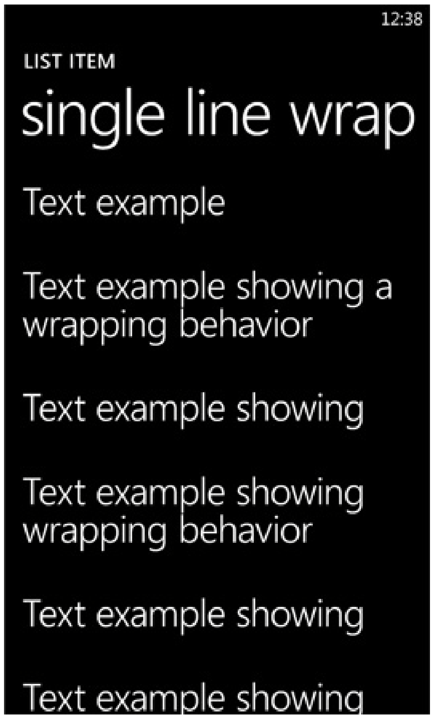
Text example

Text example showing scre

Text example

Text example showing scre

Text example



12:38

LIST ITEM

single line wrap

Text example

Text example showing a wrapping behavior

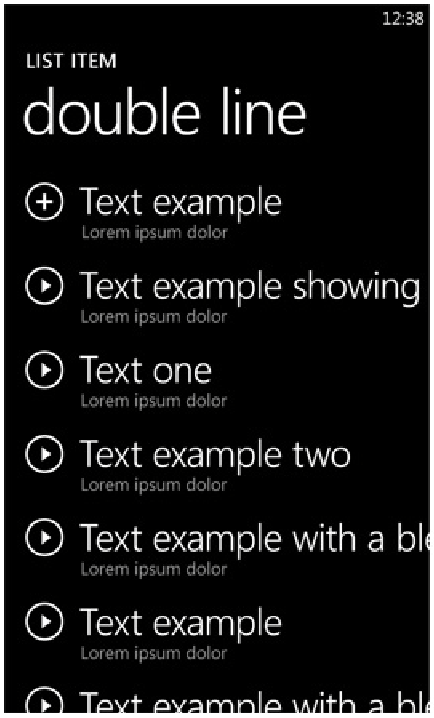
Text example showing

Text example showing wrapping behavior

Text example showing

Text example showing

A second example is the double line format with image icon. This control supports multiple lines of text with an image placeholder that can be generated to the right or left of the control boundaries.



12:38

LIST ITEM

double line

Text example
Lorem ipsum dolor

Text example showing
Lorem ipsum dolor

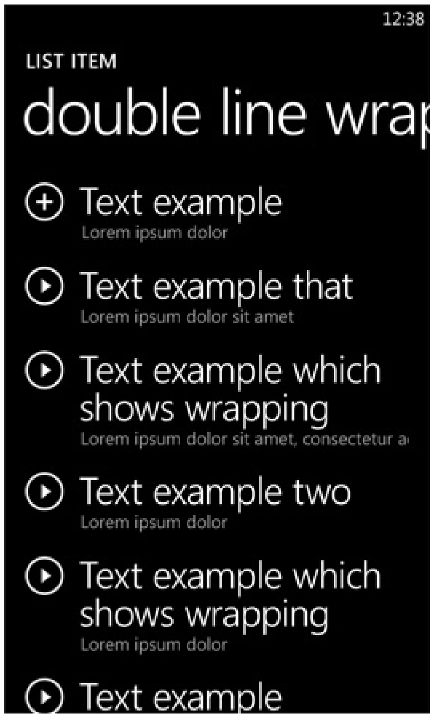
Text one
Lorem ipsum dolor

Text example two
Lorem ipsum dolor

Text example with a bl
Lorem ipsum dolor

Text example
Lorem ipsum dolor

Text example with a bl



12:38

LIST ITEM

double line wra

Text example
Lorem ipsum dolor

Text example that
Lorem ipsum dolor sit amet

Text example which shows wrapping
Lorem ipsum dolor sit amet, consectetur a

Text example two
Lorem ipsum dolor

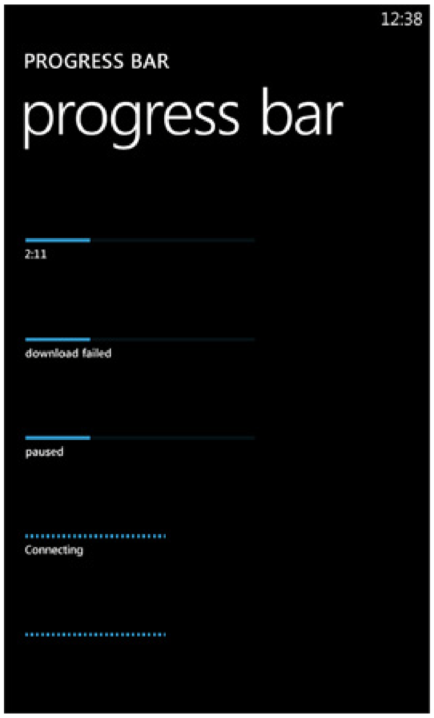
Text example which shows wrapping
Lorem ipsum dolor

Text example

List Item Design Considerations

- ✓ Multiple hit targets are only available if you use the list box control.
- ✓ When using multiple hit targets, the primary action should be larger than the secondary action.
- ✓ Primary hit targets should navigate further into the application, and secondary hit targets should perform a specific action.

Progress Bar



The progress bar is a control that indicates the progress of an operation. You can use the control to show generic progress, or the progress that changes according to a value. Some features include:

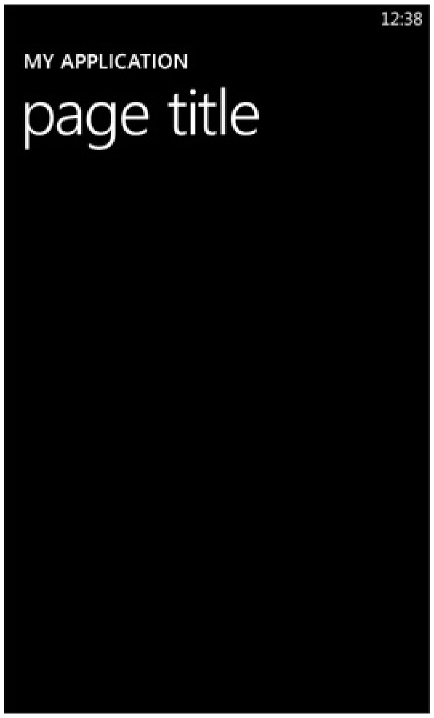
- ✓ The progress bar supports a marquee (indeterminate) mode.
- ✓ The progress bar supports theming.

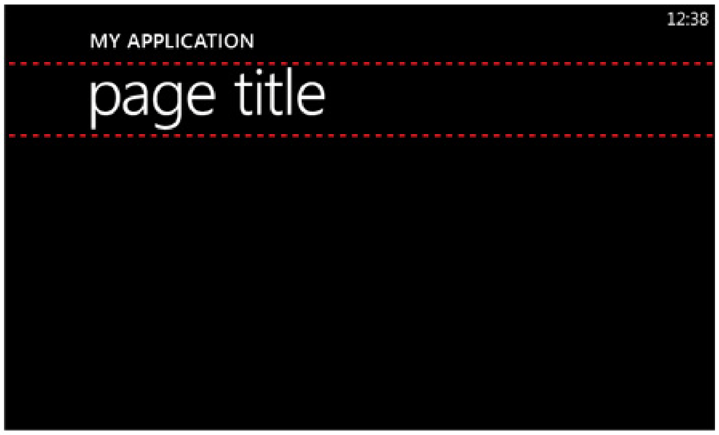
Application Design Considerations

- ✓ The progress bar has an enabled and disabled state. When it is enabled you can interact with it such as stopping progress.

Page Title

Although not a usable control, the page title layout will be discussed in this topic. Page titles are used to clearly display information for page contents.





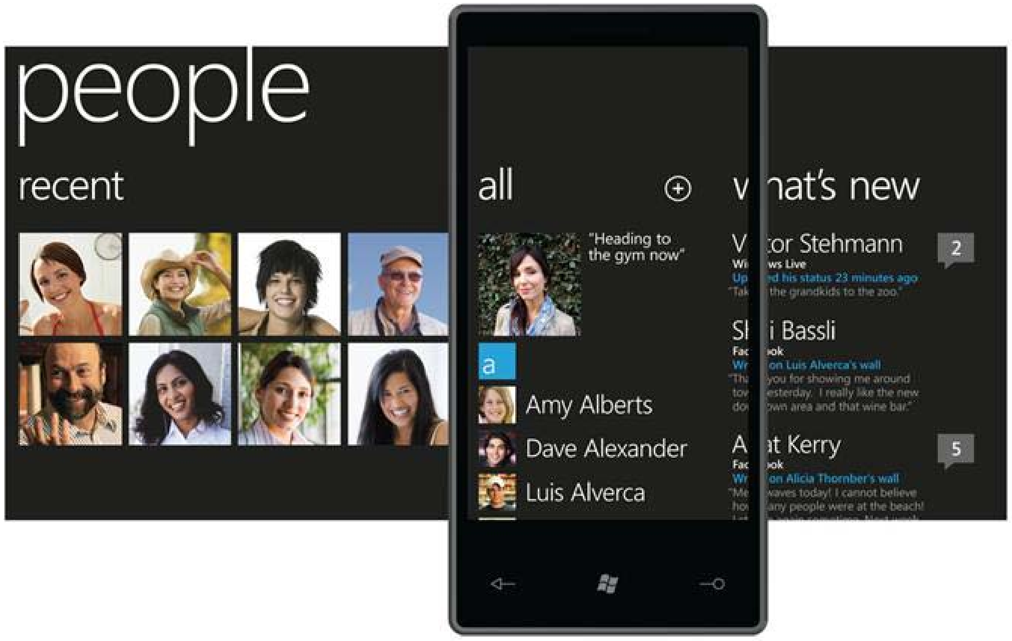
Application Design Considerations

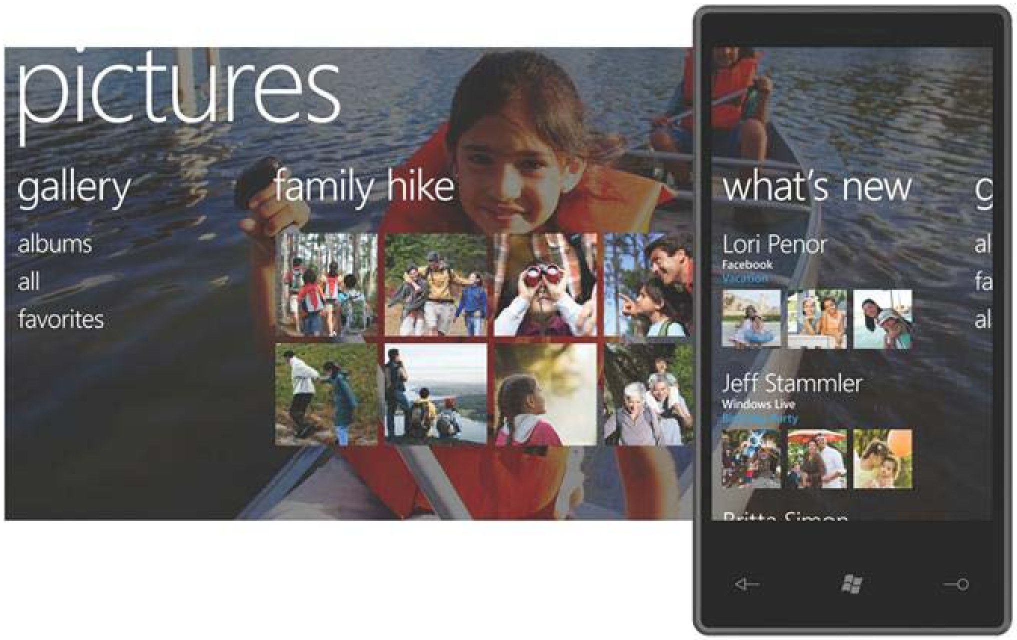
- ✓ Page titles do not scroll.
- ✓ Applications can choose to show or not show the page title. If an application chooses to display a page title, the space should be reserved for all pages in the application for consistency and the user does not experience differing window size.
- ✓ If an application displays the page title, the title should be the name of the application, or a specific descriptive line of text relevant to the displayed data. Page titles are not actionable.

Panorama Application

Panoramic applications are a part of the core Windows Phone OS 7.0 CTP experience. Unlike standard applications that are designed to fit within the confines of the phone screen, panoramic applications offer a unique way to view controls, data, and services by using a long horizontal canvas that extends beyond the confines of the screen. These inherently dynamic applications use layered animations and content so that layers smoothly pan at different speeds, similar to parallax effects.

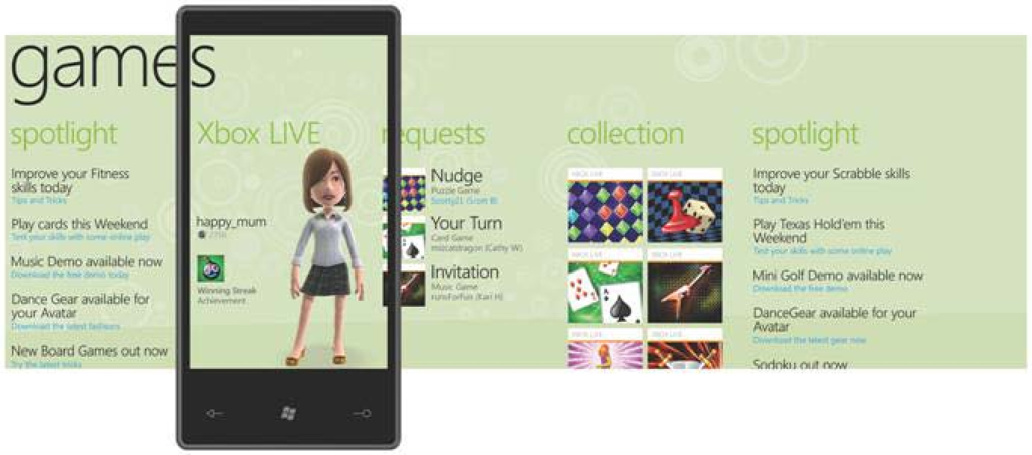
Currently, there is not a panoramic application template or control provided as part of the standard application platform. However, by using Silverlight developers can create similar application experiences.





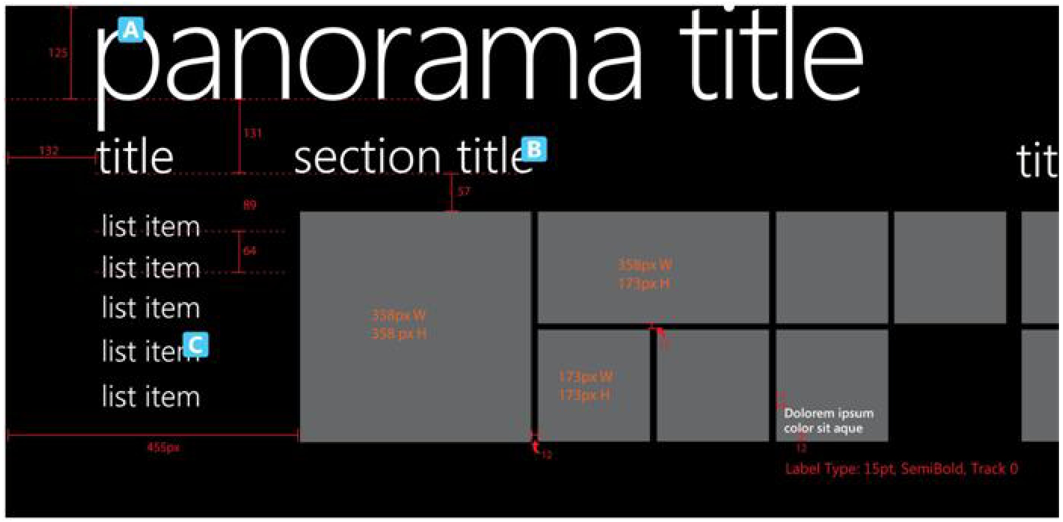
Design Recommendations

Although there are no official limitations for custom Silverlight panoramic applications, Microsoft does offer a number of recommendations. The goal of these recommendations is to help you mirror the integrated panoramic applications and provide a more uniform experience on the phone.


games
spotlight
Improve your Fitness skills today
Play cards this Weekend
Music Demo available now
Dance Gear available for your Avatar
New Board Games out now
Xbox LIVE
happy_mum
requests
Nudge
Your Turn
Invitation
collection
spotlight
Improve your Scrabble skills today
Play Texas Hold'em this Weekend
Mini Golf Demo available now
DanceGear available for your Avatar
Sodoku out now

Common UI Elements

The user interface of a panoramic application consists of four layer types that operate with their own independent motion logic: a background image, a panorama title, panorama sections, and panorama section titles.

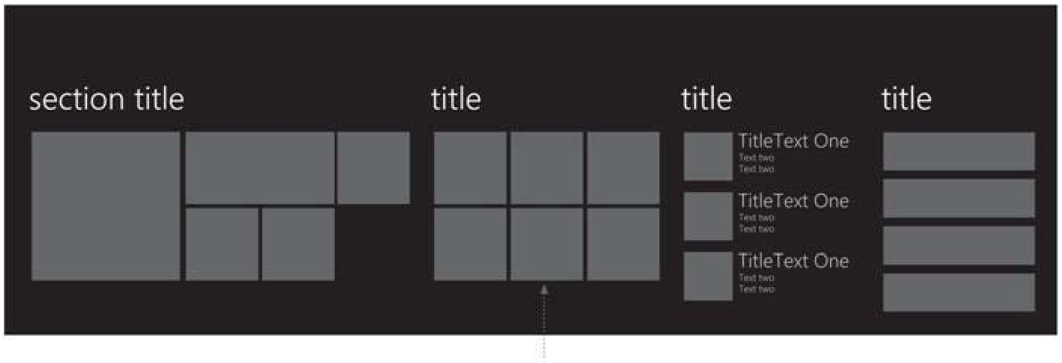


A Segoe UI Light
140 pt
-10 Tracking

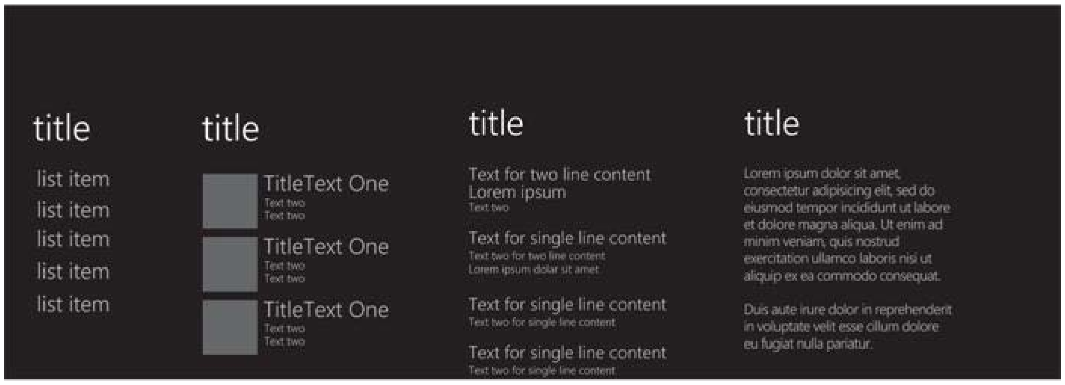
B Segoe UI Semilight
54 pt
-10 Tracking

C Segoe UI Semilight
24 pt
-10 Tracking

The following image illustrates image based elements.



The following image illustrates text based elements.



title

list item
list item
list item
list item
list item

title

TitleText One
Text two
Text two

TitleText One
Text two
Text two

TitleText One
Text two
Text two

title

Text for two line content
Lorem ipsum
Text two

Text for single line content
Text two for two line content
Lorem ipsum dolar sit amet

Text for single line content
Text two for single line content

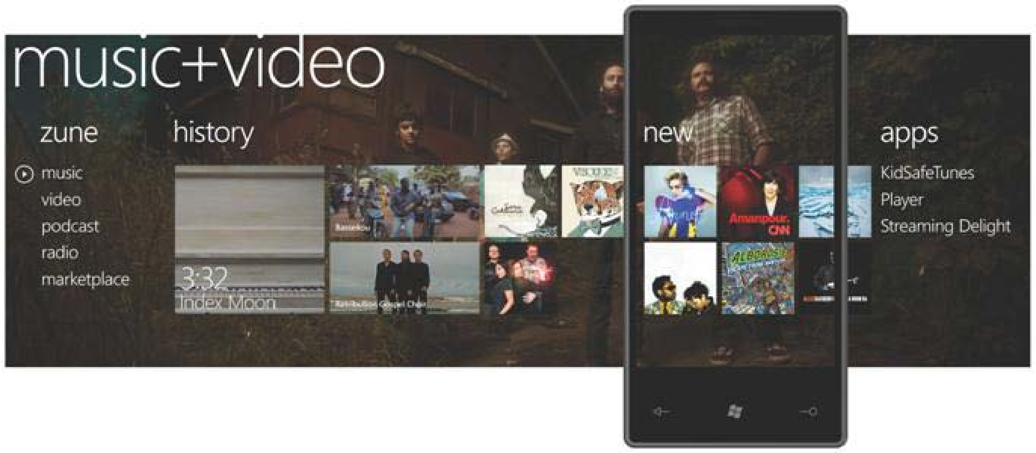
Text for single line content
Text two for single line content

title

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.

Background Image

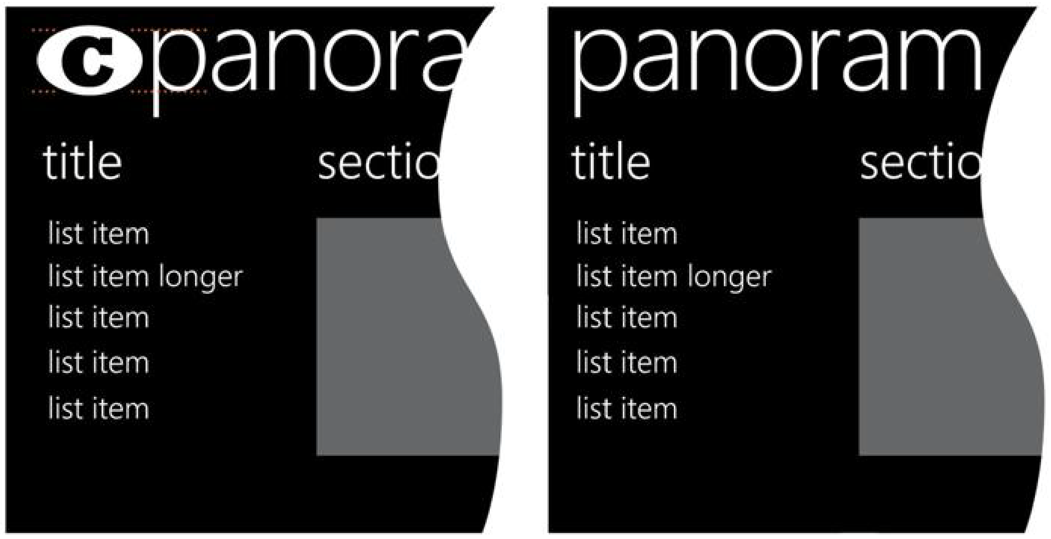


The background image is the lowest layer of a panoramic application and is meant to give the panorama its rich magazine-like feel. Usually a full-bleed image, the background is potentially the most visual part of the application, but it is important to keep a number of considerations in mind to create a great experience:

- ✓ Use either a single color background or an image that spans the entire panorama. If you decide to use an image any UI image type supported by Silverlight is acceptable, although for size considerations you might consider JPGs.
- ✓ Use of multiple images as background is fine, but only one should be displayed at any given time.
- ✓ Use an image size between 800 x 480 and 800 x 1024 pixels (height x width) to ensure good performance, minimal load time, and no scaling.
- ✓ Use a 16 x 9 aspect ratio for a panoramic application that has four sections.
- ✓ Use a transparent black or white filter to aid in text legibility.

- ✓ Avoid using drop-shadow effects on elements of the UI that are dynamic.
- ✓ Use a rate of motion that is relative to the panning gesture determined by the total width of the top content layer relative to the width of the background image.
- ✓ Use animated motion only when there is background art present in the application.
- ✓ Wrap off and then back onto the visible area when the user pans beyond the width of the image.

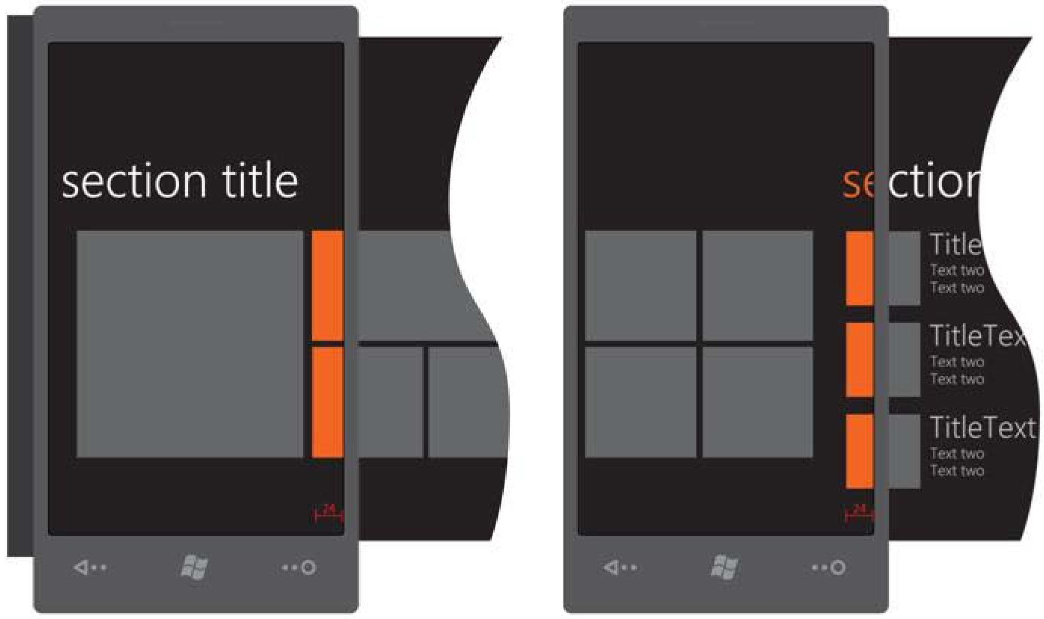
Panorama Title



The panorama title is the title of the overall panoramic application. It is meant to let the user identify the application and should be visible no matter how they enter the application. The following are recommendations for panorama titles:

- ✓ Use either plain text or images, such as a logo, for the panorama title. Using multiple elements, such as a logo and text (or other **UIElement**s) is also acceptable.
- ✓ Ensure that the font or image color works across the entire background and that the title is not dependent on the background image for visibility.
- ✓ Use the same title for the launch tile in the Start menu for consistency.
- ✓ Avoid animating the title or dynamically changing its size.
- ✓ Use a rate of motion that is slow relative to the topmost content layer, but that is faster than the background art.
- ✓ Wrap off and then back onto the visible area when the user pans beyond the width of the image.

Panorama Sections



Panorama sections are the component of the panoramic application that encapsulates other controls and content. The following are recommendations for panorama sections:

- ✓ Ensure panoramic applications perform smoothly by using a maximum of four sections.
- ✓ Vertical scrolling through a list or grid is acceptable as long as it is within the confines of the section and is not in parallel with a horizontal scroll.
- ✓ Vertical scrolling is acceptable as long as the section's width is less than the width of the screen.
- ✓ Support all custom and standard controls.
- ✓ Move at the same rate as the finger drag.
- ✓ Animate off the screen when a user navigates to a new section.
- ✓ Design the layout of sections so that a hint of the next section is visible. Providing a slight overlap makes it intuitive to users that they need to pan through the application.
- ✓ Consider hiding panorama sections until they have content to display.

Panorama Section Titles

A title is optional for a panorama section. If you provide a title, consider the following recommendations:

- ✓ Ideally use plain text, although images are possible. Using multiple elements, such as an image and text (or other **UIElement**s) is also possible.
- ✓ Ensure that the section title is not dependent on the background art.
- ✓ Avoid animated titles because the title will be moving.
- ✓ Span the entire section, even if multiple controls are present.
- ✓ Animate off the screen when a user navigates to a new section.

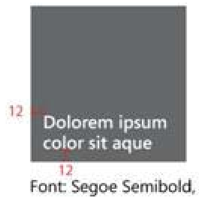
Note:

A panorama section's title should act differently depending on whether the section's width is greater than or less than the width of the screen. If the section's width is greater, there should be a horizontal animation. That is, the title should not stay in the top left of the section, but rather it should move with a different speed along the top while the panorama is moving. Under these circumstances, there should not be vertical scrolling. Alternatively, if the section's width is less than the screen width, the title should always be set to the top left of the section. Under these circumstances, there should not be any horizontal animation and, if vertical scrolling is possible, the title should move along with the content.

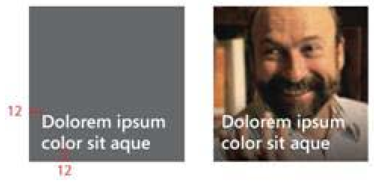
Thumbnail Specifications

Thumbnails are a main element of the panoramic view. They link to content or media that is consumed outside of the panoramic experience. As illustrated by the following image, you should use cropped images that highlight an identifiable subject rather than an entire image. If the image is not identifiable without text, up to two lines of text could be used to identify the content.

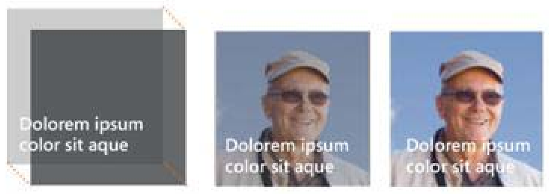
Thumbnail without system text



Font: Segoe Semibold, 15pt, Track 0



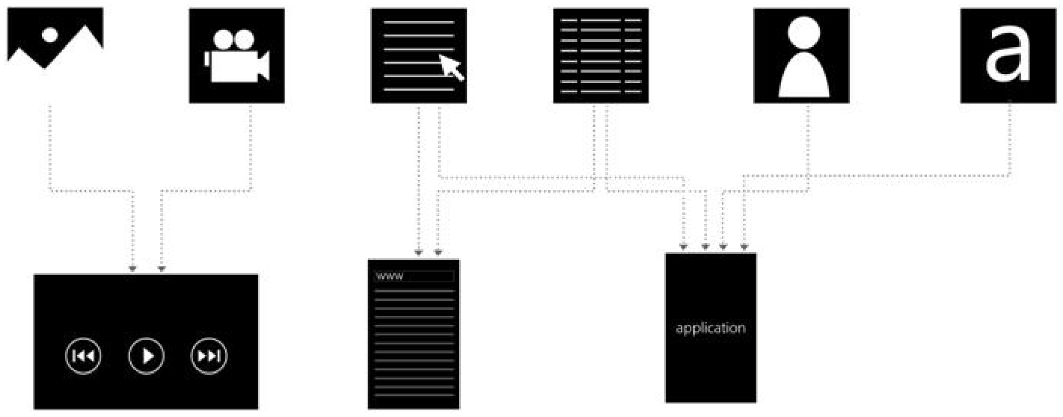
Thumbnail with text, no masking needed



Thumbnail with text + 25% black mask over picture

Element Flow

Elements of a panoramic application serve as the starting point for more detailed experiences. The following image illustrates the standard flow used by the core panoramic applications on the phone.



Note:

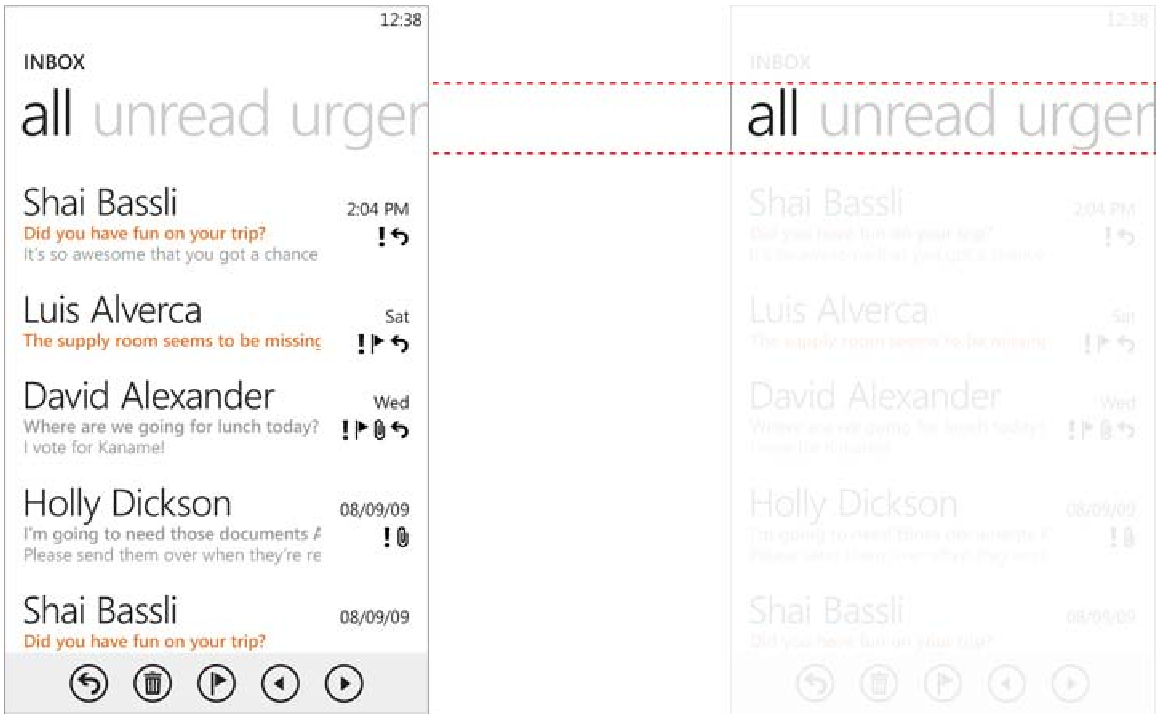
The element flow shown above is not indicative of platform functionality, but rather the end-user experience. As an example, while an application launching from a panoramic application might be what the end-user sees, the launched application is actually just a different view of the same panoramic application.

Pivot Control

A pivot control provides a quick way to manage views or pages within the application. The pivot can be used for filtering large datasets, viewing multiple data sets, or switching application views. The pivot control places individual views horizontally next to each other, and manages the left and right navigation. Flicking or panning left to right on the page advances the pivot.

Note:

Again this control is not available in the CTP release. However, the below design considerations will highlight some of the main features and considerations to take when developing your own pivot control.



Shai Bassli 2:04 PM
Did you have fun on your trip?
It's so awesome that you got a chance

Luis Alverca Sat
The supply room seems to be missing

David Alexander Wed
Where are we going for lunch today?
I vote for Kaname!

Holly Dickson 08/09/09
I'm going to need those documents A
Please send them over when they're re

Shai Bassli 08/09/09
Did you have fun on your trip?

Application Design Considerations

- ✓ Applications should minimize the number of pivot pages.
- ✓ The content of the pivot page is defined by the application.
- ✓ Pivot pages are cyclical.
- ✓ Pivot pages must not override the horizontal pan and horizontal flick functionality as it collides with the interaction design of the pivot control.
- ✓ There is no limit to the pivot header text length. The amount of text displayed is constrained by the width of the pivot control.
- ✓ The pivot header is a fixed height and cannot be changed.
- ✓ The pivot control should only be used to display items or data of similar type.
- ✓ This control should not be used for task flow. Different pages should flow seamlessly in terms of look and feel. Also pages should not change user activity drastically.
- ✓ Pivot controls should be used sparingly and limited to scenarios where it is appropriate for the experience.
- ✓ An empty pivot page should only be removed when there is no chance additional information can be added through user action.

Windows Phone System Controls

This section covers the system controls for Windows Phone 7 Series CTP. This discussion will detail on how to use these controls and/or how they will interact with your application.

- ✓ System Tray and Application Bar
- ✓ Context Menu
- ✓ Windows Phone Notifications
- ✓ Progress Indicator

System Tray and Application Bar

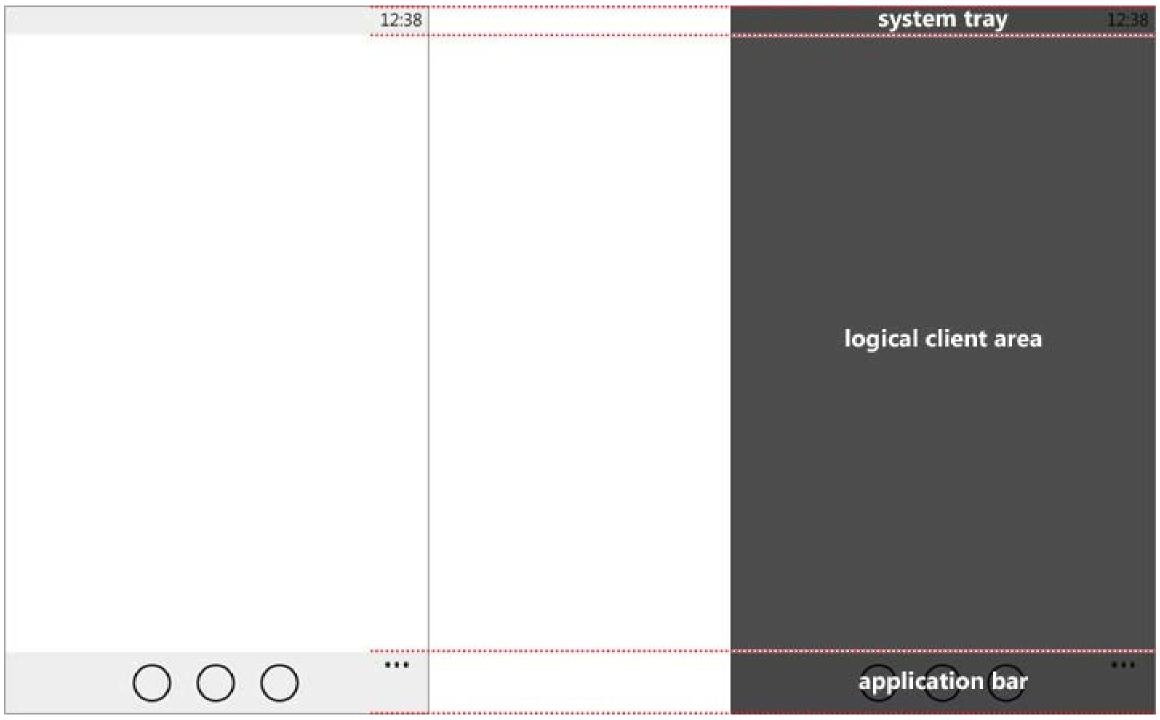
The Windows Phone 7 Series CTP chrome contains two primary components, the system tray and application bar.

- ✓ **System Tray** – An indicator bar that displays system-level status information in a simple and clean manner, reserved in the application workspace. The user may touch the system tray, as it updates to provide different notifications and keep them aware of things that are important.
- ✓ **Application Bar** - The application bar provides a place to promote the most common tasks and views. The application bar also provides a single, optional, application menu. The common tasks and views are displayed as graphical button on the toolbar and additional tasks are organized into a menu. In the following diagram, the system tray and application bar represent the application chrome.

Note:

To show or hide the system tray and application bar use the **FullScreen** property. To show or hide the application bar and change opacity set the **Visible** and **Opacity** properties respectively on the application bar.

The following illustrates the positioning of these elements within an application:



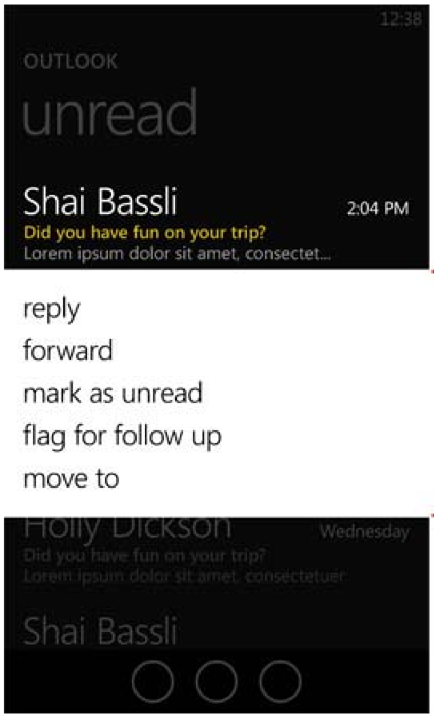
Some features of the application bar include:

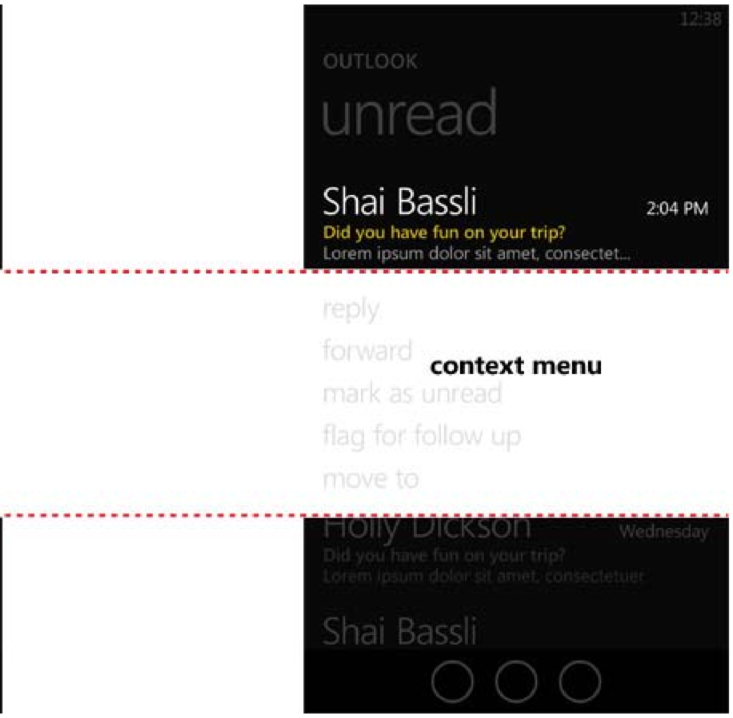
- ✓ The application bar is aligned with the bottom edge of the display and the width extends from the left to right edge of the screen.
- ✓ The application bar height is fixed and cannot change.
- ✓ Added buttons will fill the application bar in a centered fashion from left to right.
- ✓ When there are items present in the application bar menu, a visual indicator of sequential dots will appear on the application bar. You can tap these dots to bring up the application bar menu.
- ✓ The application bar buttons may be disabled. An example would be having a delete button on the application bar. You may choose to disable the button in certain scenarios where it would not be appropriate to delete some items. For example if they are read-only elements.
- ✓ Users can dismiss the application menu by tapping outside the menu.
- ✓ There is a maximum of four items that can be placed on the application bar. You should place tasks that are non-frequent in the application menu as this process is not automatic.

Application Design Considerations

- ✓ Use the default system theme colors for the application bar unless there is a compelling reason to customize the colors. Using custom colors for the application bar can affect the display quality of the button icons, can cause unusual visual effects in menu animations, and can even influence power consumption on some display types.
- ✓ The opacity of the application bar can be adjusted finely, but it is recommended that you only use opacity values of 0, .5, and 1. If the application bar opacity is less than 1, the application bar will overlay the UI. If the opacity is set to 1, the displayed page size will change.
- ✓ Gesture Support: You can flick or tap the application bar to bring up the application menu.
- ✓ Application bar icon images should be 48 x 48 pixels. Images that are other sizes will be scaled to fit, but image quality will be lower.
- ✓ Icon images should use a white foreground on a transparent background using an alpha channel. The application bar will colorize the icon according to the current style settings and colored icons can cause this effect to display unpredictably.
- ✓ The circle displayed on each icon button is drawn by the application bar and should not be included in the source image.
- ✓ Use icon buttons for the primary, most-used actions in your application view. Some actions are difficult to convey clearly with an icon. If this is the case, consider using a menu item instead. Applications should not fill all four slots if not necessary.
- ✓ Menu item text will run off the screen if it is too long. There are not a defined number of characters that will fit on the screen, but due to variable length characters text should be between 14 and 20 characters. This is the recommended maximum range for menu item text length.

Context Menu


12:38
OUTLOOK
unread
Shai Bassli
2:04 PM
Did you have fun on your trip?
Lorem ipsum dolor sit amet, consectet...
reply
forward
mark as unread
flag for follow up
move to
Holly Dickson
Wednesday
Did you have fun on your trip?
Lorem ipsum dolor sit amet, consectetuer
Shai Bassli


12:38
OUTLOOK
unread
Shai Bassli
2:04 PM
Did you have fun on your trip?
Lorem ipsum dolor sit amet, consectet...
reply
forward
context menu
mark as unread
flag for follow up
move to
Holly Dickson
Wednesday
Did you have fun on your trip?
Lorem ipsum dolor sit amet, consectetuer
Shai Bassli

The context menu offers an additional way for users to access specific tasks. To eliminate the need for scrolling, there is a maximum of 5 items allowed for the context menu.

Note:

Although this section shows the system control version of the context menu, for the current release developers will need to create their own custom context menu in their application. The below section provides some features and guidance that may prove useful in creating this type of menu.

Some features include:

- ✓ Context menus will remain on the screen until a user dismisses the menu. Other dismissal options include tapping outside the menu, choosing the hardware back button, or selection of a menu item. System events may also dismiss a context menu.
- ✓ Gesture support: You can tap and hold on a list item to bring up the context menu.

Note:

Context menus should not be the only way to accomplish a task. It is one way to accelerate a navigation process.

Windows Phone Notifications

In Windows Phone 7 Series CTP, there are several methods available to notify a user of events on the device. The types of notifications are the following:

- ✓ Awareness notifications inform users of changes or events that may have occurred and are non-disruptive to the user workflow. They will have visual cues in the start menu called tiles.
- ✓ Action requested notifications are system-wide notifications that do not disrupt the user workflow or require intervention to resolve. An example of these notifications is when the end user receives a text message or instant message.
- ✓ Action required in application notifications are fully controlled by an application and affect only the application.

Push Notification Overview

For application development, the push notification service is designed to provide a cloud service with a dedicated, resilient and persistent channel for pushing a notification to a mobile device. When a cloud service needs to send a push notification to a device, it sends a notification request to the Push Notification service, which in turn routes the notification to the application, or to the device as a toast or tile notification, depending on the type of notification that is sent. The Push Client on the device receives push notifications through a notification channel. When the channel is created, a subscription is also created which allows the cloud service to push notifications to that channel. A channel is represented by a URI which contains all of the information associated with the subscription. Once an application receives the push notification, it can access the cloud service using the cloud service's protocol to retrieve any needed information.

Push Notification Types

Tiles and toast notifications are two mechanisms that enable a cloud service to deliver relevant, actionable feedback to users outside an application's own user interface. Additionally, a cloud service can send a raw notification request. Depending on the type of notification sent, the notification will be routed either to the application or the shell.

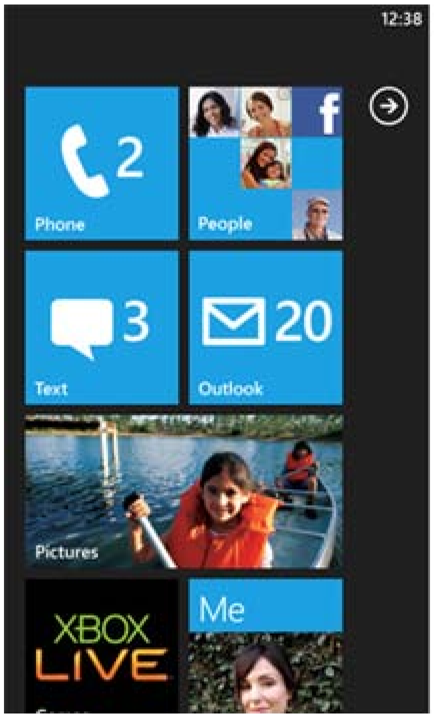
Tile Notifications

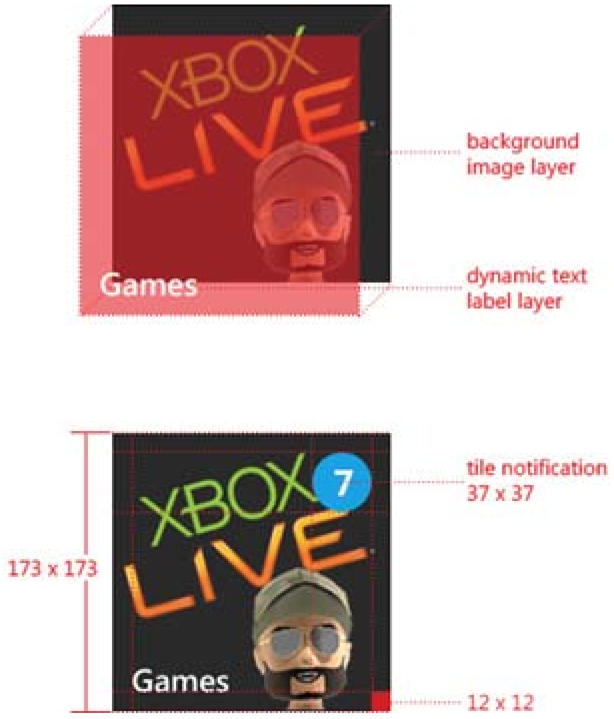
A tile is a visual, dynamic representation of an application or its content within the quick launch area on the phone start experience. For example, a weather application may choose to display the local time and climate conditions in a tile. Because a cloud service can alter the tile appearance at any time, this mechanism can be used to communicate information to the user on an ongoing basis. Each application that the user can launch on the phone is associated with a single tile, but only the user can control which of these tiles are pinned to the quick launch area. Currently there is no way for an application to know if its tile has been pinned to the quick launch area.

A cloud service can control a tile's background image, counter (or 'badge'), and title properties. These properties will be configured using the Windows Phone Developer Tools. Animation and sound properties are controlled by how the platform is configured, not by the application. For example, if the platform is configured to animate and beep upon any tile update, that is what will occur for any tile.

A tile's background image can reference either a local resource, which is part of the application deployment, or a cloud resource. By referencing a resource in the cloud, applications are enabled to dynamically update a tile's background image. This enables scenarios which require processing of the background image before it is displayed. In most scenarios, the application package should include all needed background images for the tile, since this is the best solution for performance and battery life.

The following represent the tile anatomy and tile image dimensions needed when constructing the tile UI:





Note:

Tile images can either be in JPG or PNG formats. The tile notification counter is an optional component in tile construction. For more information on how to create and modify a tile see: http://go.microsoft.com/fwlink/?LinkId=185182.

Toast Notifications

A cloud service can generate a special kind of push notification known as a toast notification, which displays as an overlay onto the user's current screen. For example, toast notifications that are produced via an instant messaging client, or another peer to peer oriented application. If the user decides to click the toast notification, the application can launch and perform other actions. A cloud service can control a toast notification's title and sub-title. The toast notification will also display the application's icon that is included in the application's deployment package.

The following should be noted about toast notifications:

- ✓ Toast notifications should be personally relevant and time critical.
- ✓ They should primarily be focused on peer to peer communication.
- ✓ Applications must default to turn toast notifications off. The user must opt-in explicitly in the application UI or settings menu to enable toast notifications.

Raw Notifications

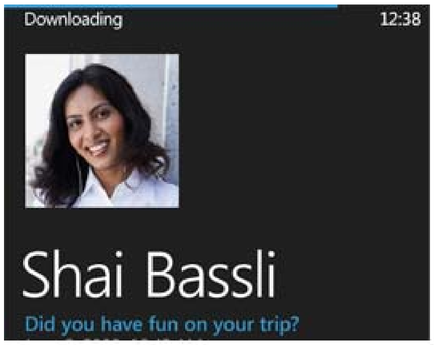
If an application wants to receive messages from its cloud service without using push notifications, it would have to constantly poll the cloud service (using HTTP long polling). By using the Push Notification service, an application can instead receive raw notifications from its cloud service while it is running in the foreground. This eliminates the need for an application to write custom code to keep a persistent connection open with its back end cloud service. If the application is not running in the foreground, the raw notification is discarded on the Push Notification service and is not sent to the device.

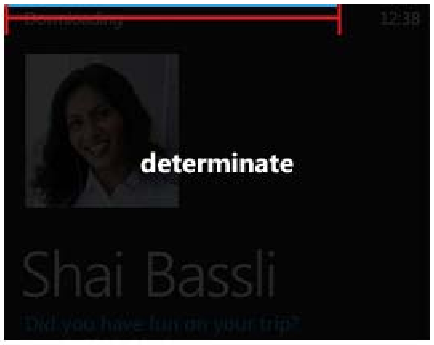
Progress Indicator

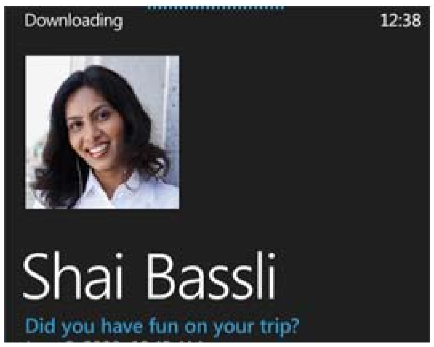
The progress indicator shows in-application activity related to an activity or a series of events. This system control is integrated into the system tray and can display across multiple application pages.

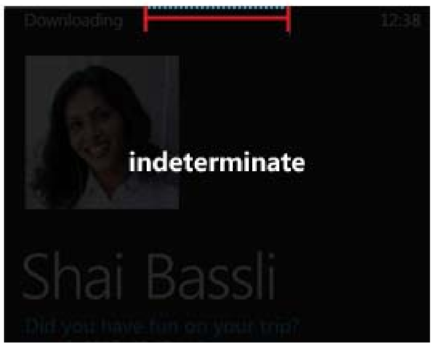
The progress indicator may be either determinate or indeterminate:

- ✓ Determinate has a beginning and end point.
- ✓ Indeterminate continues until a task is finished.









UI Text Guidelines

This topic provides the recommended UI text guidelines for Windows Phone 7 Series CTP. Although optional, the following guidelines may help to create an informed, cleaner, and friendlier text experience for your application. If desired, many of the suggestions here will help your UI parallel the native Windows Phone 7 Series CTP text format.

Windows Phone 7 Series Voice and Tone

This section provides insight into the voice and tone within a Windows Phone 7 Series CTP application. It serves as an example as these elements can vary from application to application. However, a developer should be aware of the voice and tone and how it is projected in the UI text. Also, they should maintain a level of consistency with voice and tone throughout their application.

Voice refers to the personality within the text. For example, the voice of the writer would be their overall personality incorporated into what they write. Ideally, the voice of Windows Phone 7 Series CTP applications should be authentic, clear, and reflective of the language used by the audience. Tone is the overall mood of the text such as happy or angry. For applications, we would suggest friendly, lighthearted, and empathetic tones. Never use an angry or mechanical tone in the application.

An example would be to use the voice and tone of a friend assisting another friend with something on the phone. The scenario might be helping them understand an error message that appears in the application. A developer shouldn't offer a rigid, uninformative response when trying to explain an issue. For example, most end users will not relate to **Error Code: 4560363**. However, something such as **There is some info missing here. Please enter your name in the text box to move to the next page."** will work. It is imperative to give them a meaningful response in a casual, comprehensible manner. Help them fix the problem in a way that they can understand.

Other examples include:

- ✓ "Synchronize the phone device". You would never tell a friend to do this. Instead use something like "sync your phone" as an alternative.
- ✓ "Schedule a calendar event for tomorrow through Outlook." Again, it is a poor example of text that is neither friendly nor representative of how a friend would speak. An alternative would be "Hey, you have to set up an appointment tomorrow in Outlook."

Although the above are simple examples, they show the importance of keeping the voice and tone personal rather than mechanical and rigid.

Capitalization

It is good to maintain consistent capitalization practices in your application to prevent a disjointed reading experience. The Windows Phone 7 Series CTP capitalization guidelines use lowercase and all caps layouts in many places, but also the following:

- ✓ **Title Caps** – This is where you capitalize the first and last words of the phrase and all words in between. The exceptions are articles (a, an, the), coordinating conjunctions (and, but, for, not, or, so, yet), and prepositions with four or fewer letters (at, for, into). An example would be **Vitamins in My California Raisins**.
- ✓ **Sentence Caps** – You should capitalize only the first word of a phrase. The exceptions are words that would be normally capitalized in text such as proper nouns or feature names. An example would be **Vitamins in my California raisins.**

For Windows Phone 7 Series CTP, the capitalization guidelines are:

All lowercase:

- ✓ List titles
- ✓ List group titles
- ✓ Push button control text or words that function as commands
- ✓ List items
- ✓ Example text that appears in a search box
- ✓ Link controls in the middle of the sentence
- ✓ Page title in your application

Sentence caps:

- ✓ Check box and radio button labels
- ✓ Progress indicator
- ✓ Status, notification, and explanatory text
- ✓ Toggle switch

All caps:

- ✓ Application title
- ✓ Date and time
- ✓ AM or PM

Punctuation

The following table shows the standard rules of punctuation for UI elements.

Punctuation Mark	**Usage Guidelines**
Ampersand (&)	Okay to use in settings or menu lists, for example, Date & Time; Clocks & Alarms.
Colon (:)	✓ Do not use a colon at the end of labels for controls such as text boxes, drop-down lists, and progress bars. ✓ Do not use a colon when the text box or drop-down list is embedded in a sentence or when the drop-down list appears in a main window. ✓ Do not use a colon at the end of group headings or column headings. ✓ Use a colon to introduce numbers or other variables for example: Percent Fragmentation: XX%
Comma (,)	✓ In a series that consists of three or more elements, separate the elements with commas. ✓ Always use a comma before the last item in a series of three or more elements. For example: "The car was red, small, and fast."
Ellipsis (…)	✓ Use an ellipsis in progress indicator labels to indicate a continuing action, for example, when the user is downloading a file. Even if there is a visual of a progress indicator, you will still want to use the ellipsis. ✓ Do not use in headings. ✓ Do not use in button labels.

Punctuation Mark	Usage Guidelines
End punctuation (. ? !)	✓ Use ending punctuation only instructional text in the UI. Do not use end punctuation if instructional text appears in a title bar or button. ✓ Do not use a period at the end of option or check box text labels, even if the label is a sentence. ✓ Separate sentences with one space after the ending punctuation, not with two spaces. ✓ End a question with a question mark. But in general, avoid phrasing labels as questions. ✓ It is okay to use a question mark at the end of a title for an error message or dialog box.
Parenthesis ()	Avoid in the UI if possible, but okay to use if you need to include an acronym or other short piece of information.

UI Text Design Considerations

- ✓ Understand that proper use of voice and tone will create a more engaging experience for the end user. The developer should craft their application in a manner that provides clear and friendly text in UI interfaces.
- ✓ Well-defined capitalization guidelines will give your application consistency and prevent a jagged end user reading experience.
- ✓ It is imperative to use proper punctuation correctly in UI strings. Again, consistency should be of major focus within your application.