Windows Phone

UI Design and Interaction Guide for Windows Phone 7

7

July 2010
Version 2.0

UI Design and Interaction Guide for Windows Phone 7

July 2010
Version 2.0

The Windows Phone 7 design philosophy

Windows® Phone 7 is for Life Maximizers, people who are busy personally and professionally, constantly juggling priorities, and who value technology as a means to an end, a way to get things done.

They do not want to feel overwhelmed because they have priorities to balance as they grow personally and professionally, all the while living life to its fullest.

Applications should embody the three Red Threads of Windows Phone 7:

- Personal – your day, your way
- Relevant – your people, your location
- Connected – your stuff, your peace of mind

Every application should connect to at least one of these threads. Create applications that can be personalized by humanizing them to display people whom the users know or places that users want to go to, and make it easy to share information across the web and beyond.

Build authentic experiences.

The Windows Phone 7 design philosophy

The Windows Phone OS 7 User Interface (UI) is based on a design that is internally named Metro, and echoes the visual language of airport and metro system signage in its design and typeface. The goal is to create contextual relevance through content – the user's own content – so that using the phone is a personal experience. Metro design interfaces embody harmonious, functional, and attractive visual elements that encourage playful exploration so that the user feels a sense of wonder and excitement. A clear, straightforward design not only makes an application legible, it also encourages usage and can lead to delight.

The Metro design was developed using the five following principles:

1) Clean, light, open, and fast: It is visually distinctive, contains ample white space, reduces clutter and elevates typography as a key design element.

2) Content, not chrome: It accentuates focus on the content that the user cares most about, making the product simple and ap proachable for everyone.

3) Integrated hardware and software: Hardware and software blend into each other and creates a seamless user experience from single-button access to Search, Start, Back and the camera to on-board sensor integration..

4) World-class motion: The Windows Phone 7 touch and gesture experiences on capacitive screens are consistent with Windows 7 on the desktop and include hardware-accelerated animations and transitions to enhance the user's experience at every turn.

5) Soulful and alive: A personalized, automatically updated view into the information that matters most to the user is enabled and brings to life a cinematic photo and video experience by having a fully integrated Zune media player experience.

These design principles are based around the concept that UI elements should be authentically digital and embody harmonious, functional, and attractive visual elements. Applications should engage users by promoting navigation, exploration, and exciting visuals in their design.

The Windows Phone 7 design philosophy

Developers should use digital metaphors where natural and appropriate and should not necessarily try to mimic real world interaction if it is not appropriate. If it is, the UI should look and feel great even though the UI objects only visually imitate and mimic analog manipulation behaviors. The Windows Phone Developer Tools provides a collection of Metro-inspired Silverlight controls for use in applications.

Microsoft highly recommends that Windows Phone 7 developers adopt the Metro design style for their applications. This guide provides the design knowledge, fundamentals, and guidance to do so. Although requirements and implementations will vary from application to application, utilizing Metro styled elements will create a more consistent and fluid overall UI experience for users.

This guide also details the methods of user interaction that can be used by a Windows Phone 7 application, including standard input, functionality within the UI framework, and the Metro-inspired Silverlight® and system-based controls. Diverging from the Windows Phone 7 interaction model is generally not allowed, but developers can gain a deeper understanding of the hardware and software interaction elements that are available as a part of the development platform, and those that are customizable.

The Windows Phone 7 human/computer interface

The first computing devices were manual objects that required touch to operate them. The interfaces were the stones of an abacus or the dials on a difference engine. The information was simply seen as the state of the device. As electronic computers were birthed, input methods rapidly evolved from switches to keyboard and mouse, and output methods from silent, blinking lights to high-definition displays with stereo sound. This transition exponentially increased what could be done with computers, but ironically made them much harder to use because the only way to manipulate them or their data was through an interface that was only a proxy to the computing event hidden away inside a case.

While a child can play with an abacus and intuitively learn how to operate it through exploration and play, the same cannot be said for computing devices that do not have a touch component.

The Windows Phone 7 UI is designed around touch interaction, offering full navigation using a combination of finger gesture movements. Knowledge about interacting with layouts should be inherently obvious. There are no complex key chords or arcane commands to memorize, and the keyboard's role is to input text. People intuitively tap, flick, pan, and otherwise touch and manipulate content directly since the content is the interface.

With A-GPS functionality, sensors such as an accelerometer, and a vibration unit, the UI can be extended beyond the surface of the phone to include where it is in the world, what orientation it is at, and how it feels – the phone itself is the UI.

Usability and the UI should be a primary design goal in every application for Windows Phone 7, not an afterthought.

As developers create applications, they should place special emphasis throughout to ensure that layouts, pictures, visual elements and touch-based controls fit this UI paradigm.

Celebrate and elevate the content to be the experience by using the UI to create awesome, unique applications that draw people in and encourage touch.

A note on units: pixels vs millimeters

All Windows Phone 7 phones will have WVGA screens at 800 x 480 pixel resolution, no matter the screen size. Most of the measurement units in this guide are expressed in pixels but in certain cases, usually around touch target size, measurements may be expressed in millimeters.

Since these units are not directly convertible without knowing the pixels per millimeter of a given screen, designers and developers who require fine-grained millimeter positioning or sizing of elements for a given screen size will need to refer to original equipment manufacturers display specifications as there is no method to determine this programmatically.

All of the controls and UI elements within the Windows Phone Developer Tools are sized to support all possible screen sizes for Windows Phone and adhere to minimum millimeter touch targets regardless of the screen size.

If developers or designers require precise millimeter sizing for UI elements, consult original equipment manufacturers display specifications for the proper conversion factor to go from pixels to millimeters.

A note on game UI design

Games are naturally immersive environments and their UIs should flex to accommodate the needs of the game.

By creating games that are designed from the beginning to use a multitouch screen, games will make the most of the primary input device of the phone, and control systems will feel natural to Windows Phone users. Though there are hardware buttons on the device, only the Back button is available to the game, and Back should only be used for the specific purpose of pausing and exiting the game.

Think about what types of control schemes fit well with a multitouch device, and break away from simulating traditional controls, such as thumbsticks as they take away useful space from the gameplay area. Instead, use gestures, such as point, stretch, shrink, flick, and turn as user input instead. Allow players to draw paths on the screen to direct units and issue commands; allow them to select groups of units by stretching an on-screen rectangle around them. Allow players to navigate by dragging the landscape with a swipe gesture, or to rotate the view by turning it with two fingers. There are many possibilities for game control using touch, and by choosing a scheme that seems natural and intuitive to gamers, you'll provide the best experience on Windows Phone.

For full-screen games, developers are free to implement whatever in-game UI elements they see fit. For games that appear within the Windows Phone page frame, developers should follow the relevant UI guidance in the rest of this document.

Visual design resources and feedback

To help designers and developers create high-fidelity visual mockups of applications that are true to the Metro design, Microsoft has created two visual design resources for inspiration and project work.

The first is the **Windows Phone Design System – Codename Metro**, a PDF book that visually explains the inspiration behind the Metro design and puts a face to the life maximizers the phone was designed for.

The second is the **Windows Phone Design Templates**, layered Photoshop template files for controls that ship as a part of the Windows Phone Developer Tools and can be used to create pixel-perfect application layouts to guide development or pitch an idea. The design templates also include examples of controls that are a part of Windows Phone OS 7.0, but are not available as a part of the Windows Phone Developer Tools. These additional templates are included to help designers and developers maintain a consistent look and feel across applications for system controls that developers wish to mimic.

The above resources and links to programming topics related to the UI elements discussed in this guide can be accessed at **http://go.microsoft.com/fwlink/?LinkID=190696**.

Microsoft values feedback on this guide and the Visual Design Resources we have made available to help developers and designers create beautiful Windows Phone 7 applications. If you have suggestions or feedback about these resources, please email us at **wp7des@microsoft.com**. We may not be able to respond to every email, but we will consider incorporating your feedback into the next versions of the resources.

Globalization and localization considerations

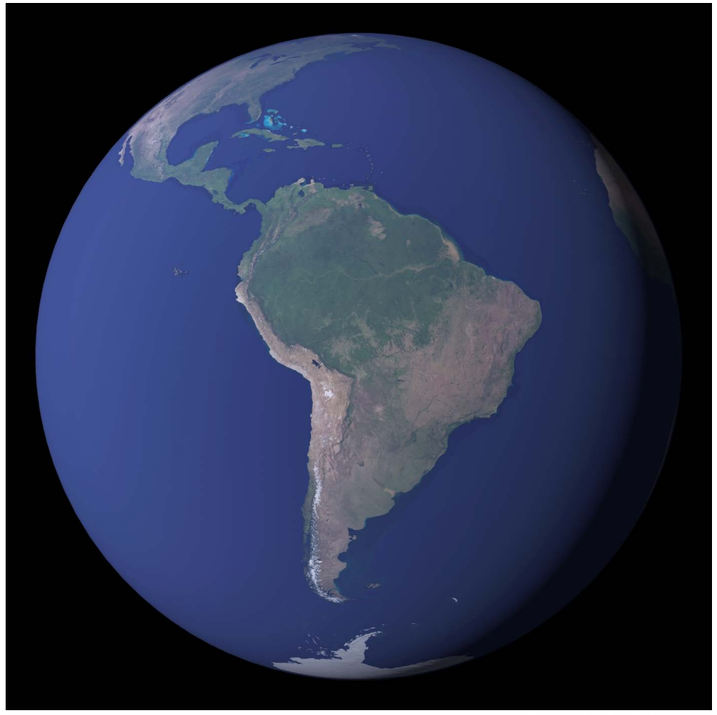
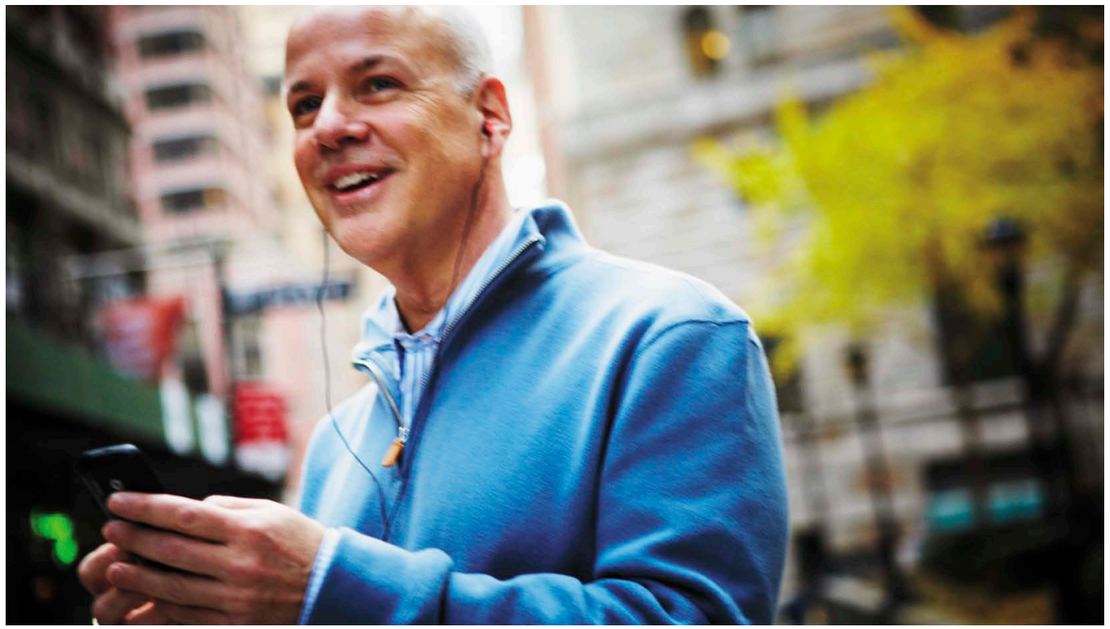


Image Source: NASA's Earth Observatory

Windows Phone 7 will be available in a number of languages and regions around the world. Developers who are interested in selling their applications to a global market should pay particular attention to making sure that their applications are world-ready by following best practices around designing the application UI to support varying text string lengths, date formats, and be aware of cultural sensitivities around use of color and images, and geopolitical issues. MSDN online, http://msdn.microsoft.com, has a variety of topics that detail these best practices.

Provide at least 40% buffer space for localized strings.

User interface framework



The Windows Phone 7 user interface framework provides consistent system objects, events, and interactions for developers and designers to create beautiful, predictable application experiences for the end user.

This section examines each piece of the framework and discusses how they can be used or accommodated within application user interfaces.

Start

Start is the beginning of the Windows Phone 7 experience for users after they power on their phone. Start displays application Tiles that users have pinned and placed in a position of their choosing for quick launch. Pressing the Start Button on the phone always returns a user to Start, no matter what application is running.

Tiles that use the Tile Notification feature can update the Tile graphic or title text, or increment a counter, enabling users to create a personalized Start experience. Examples include displaying if it is their turn in a game, the weather, or how many email messages they have received.

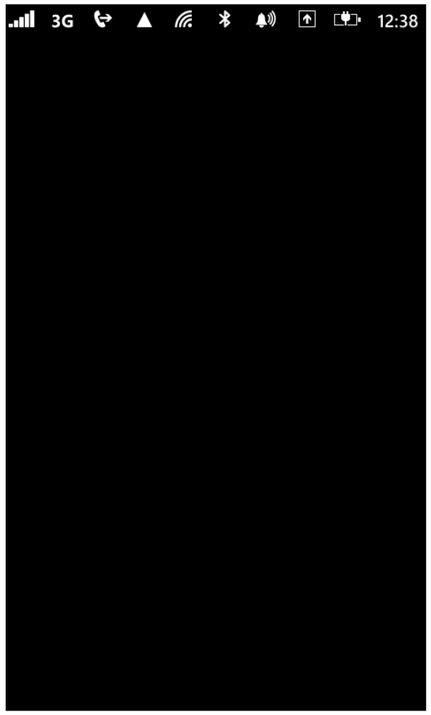
Start is always presented in portrait view.

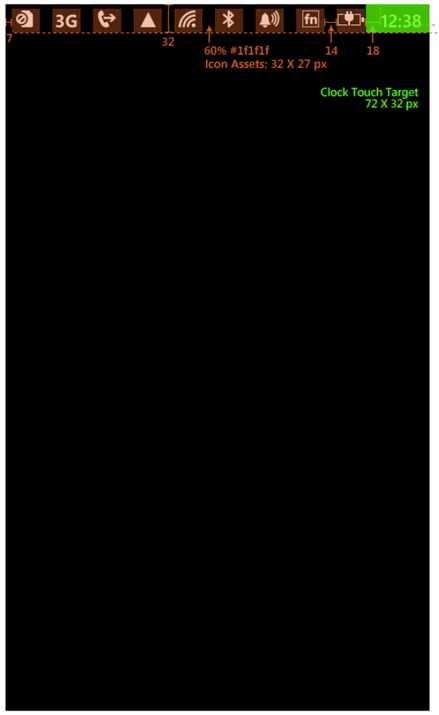
Start is a reserved space and only users can place tiles in this area. Windows Phones come with pre-placed tiles installed by Microsoft, phone manufacturers, and phone service providers.

Start is the likely to be the most viewed phone interface by users; therefore, developers and designers should carefully consider the potential that users may pin and display the Application Tile for their application in Start.

For more information, see the Application Tiles and Tile Notifications and Start Button sections.

Status bar





The Status Bar is one of the two primary components of Windows Phone OS 7.0 chrome. The Application Bar is the other.

It is an indicator bar that displays system-level status information in a simple and clean presentation in a reserved space in the application work-space. It automatically updates to provide different notifications and keeps users aware of system-level status by displaying the following information (in order from left to right):

1) Signal strength

2) Data connection

3) Call forwarding

4) Roaming

5) Wireless network signal strength

6) Bluetooth status

7) Ringer mode

8) Input status

9) Battery power level

10) System clock

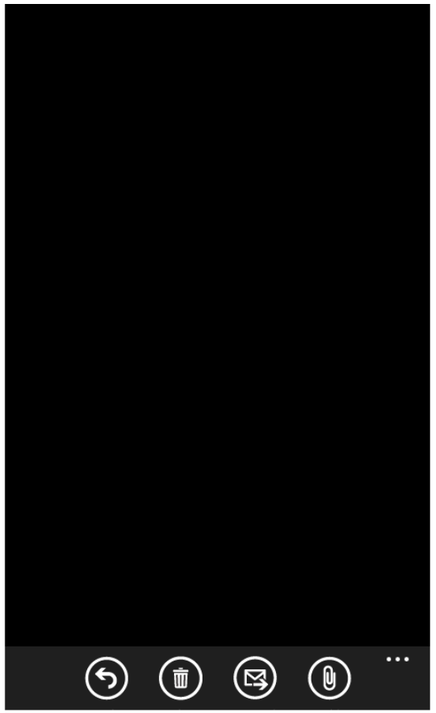
By default, only the system clock is always visible. If a user double taps in the Status Bar area, all other relevant indicators slide into view for approxi-mately eight seconds before sliding out of view.

The system clock is 32 pixels high in portrait mode and 72 pixels wide in landscape mode. It always extends to the edge of the screen and is opaque in appearance.

The Status Bar is system-reserved and cannot be modified.

It can be hidden, but many users view the system clock as an essential feature so think carefully before hiding it.

Application bar



The Application Bar provides a place for developers to display up to four of the most common application tasks and views as icon buttons.

The Application Bar provides a view that displays icon buttons with text hints and an optional context menu when a user taps the visual indicator of sequential dots or flicks up the Application Bar. This view can be dismissed by tapping outside of the menu area or on the dots, using the back button, or selecting a menu item or Application Bar Icon.

The Application Bar always stays on the same edge of the display as the Steering Buttons (Back, Start, and Search) and extends the full width of the screen in portrait or landscape mode. Icon buttons will rotate to align with the three phone orientations.

Application Bar buttons can be displayed in an enabled or disabled state. An example of a disabled button would be a delete button in read-only scenarios.

The application bar height in portrait mode and width in landscape mode is fixed at 72 pixels and cannot be modified. It can be set to be displayed or hidden.

Use icon buttons for the primary, most-used actions in the application. Do not use four icons just to use them. Less is more in this space.

Some actions are difficult to clearly convey with an icon. Place it in the Application Bar Menu instead.

For guidelines about icon button sizing, color, formatting, and text hints, see the Application Bar Icons topic. For guidelines about the Application Bar Menu, see the Application Bar Menu topic.

No text-only buttons are permitted.

Place tasks that are not frequently accessed in the Application Bar Menu.

Application Bar Menu item text will run off the screen if it is too long. The recommended maximum length for the text of a menu item is between 14 to 20 characters. Again, less is more in this space.

Use the user-defined system theme color unless there is a compelling reason to override it. Using a custom color can affect the display quality of the button icons, create unusual visual effects in menu animations, and negatively influence power consumption on some display types.

The opacity of the Application Bar can be adjusted finely, but it is recommended that you use only opacity values of 0, .5, and 1. If the opacity is set to less than 1, the Application Bar will overlay the UI. If the opacity is set to 1, the displayed page size will change.

Application bar icons


NEW


ADD

MINUS


CHECK


CLOSE


CANCEL


SYNC


REFRESH


QUESTION MARK

EXCLAMATION


BACK


NEXT


SEARCH


DELETE


SAVE


FOLDER

Application Bar Icons should be clear, understandable, and leverage real-world metaphors that are familiar to users.

The best icons have simple geometry and limit the amount of fine detail.

Icon text hints are displayed when users display the Application Bar Menu.

Use icon buttons for the primary, most-used actions in an application. Do not use four icons just to use them. Less is more in this space.

Some actions are difficult to clearly convey with an icon. Present those actions in the Application Bar Menu instead. For more information about the Application Bar Menu, see the Application Bar topic.

Application Bar Icon images should be 48 pixels by 48 pixels and have 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.

Images that are sized at sizes other than the recommended size will be scaled to fit and can potentially lower the overall image quality of the Application Bar Icon.

The circle that is displayed on each icon button is drawn by the Application Bar and should not be included in the source image.

Application bar icons

VIDEO

CAMERA

E-MAIL

SETTINGS

FAVORITES

ADD TO FAVORITES

DOWNLOAD

UPLOAD

SHARE

EDIT

OVERFLOW DOTS

PLAY

PAUSE

PREVIOUS/REWIND

NEXT/FASTFORWARD

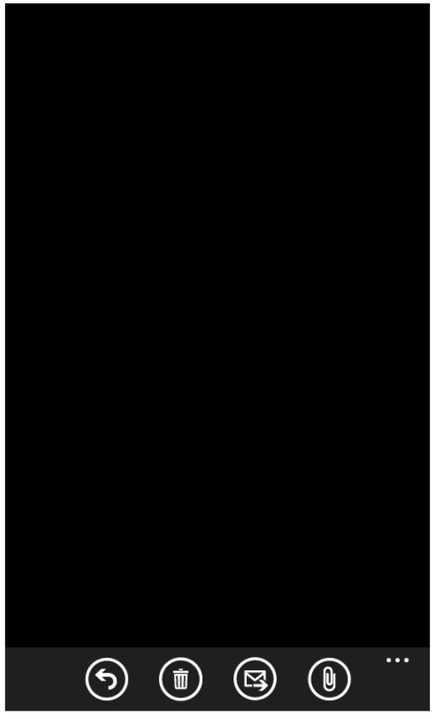
Use the user-defined system theme color unless there is a compelling reason to override it. Using a custom color can affect the display quality of the button icons, create unusual visual effects in menu animations, and negatively influence power consumption on some display types.

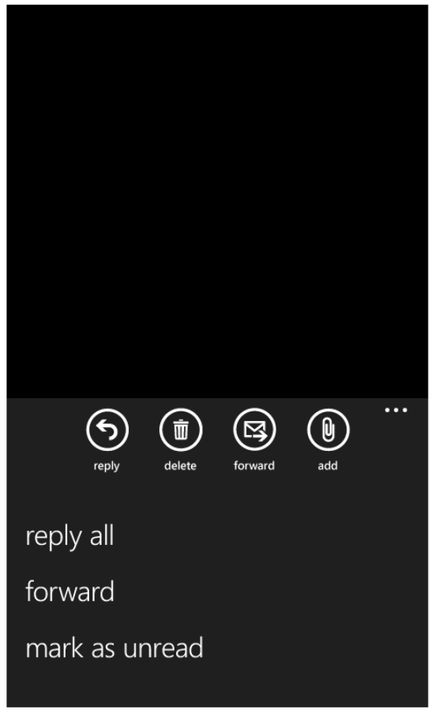
Buttons must have an icon and must have a text hint. Text hints should be short and provide context for what the button does and do not need to be fully descriptive. An example would be a button that flips an image horizontally. Instead of "flip horizontally", "flip" would be sufficient.

For more information, see the Application Bar and Application Bar Menu topic.

A set of 64 Application Bar Icons, 32 dark and 32 light in PNG format, are installed as a part of the Windows Phone Developer Tools Beta at C:\Program Files\Microsoft SDKs\Windows Phone\v7.0\Icons. Only the white icons should be used in the Application Bar.

Application bar menu





The Application Bar Menu is an optional way for users to access specific tasks from the Application Bar. The Application Bar Menu can be accessed by tapping the visual indicator of sequential dots in the Application Bar or by flicking the Application Bar up. This view can be dismissed by tapping outside of the menu area or on the dots, using the back button, or selecting a menu item or Application Bar Icon.

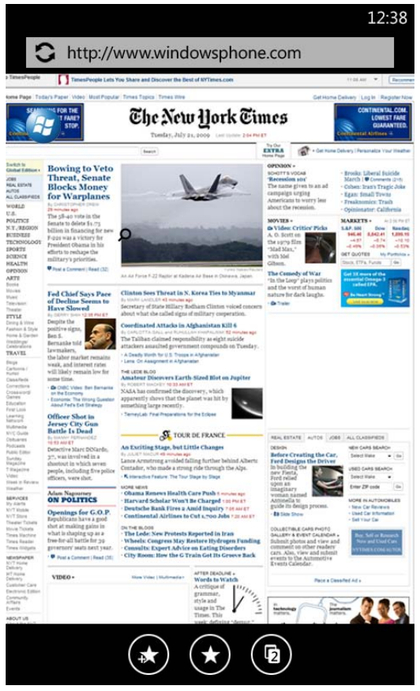
To prevent the menu from scrolling, the number of items displayed in the menu is limited to five.

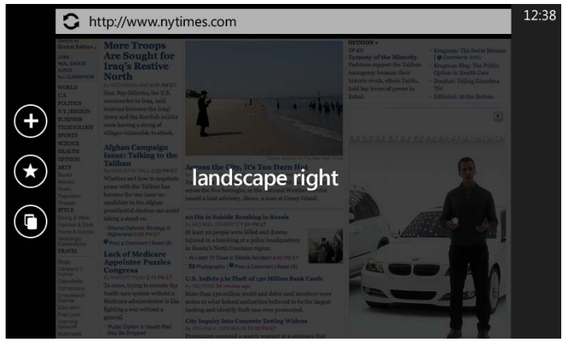
A maximum of five menu items can be displayed.

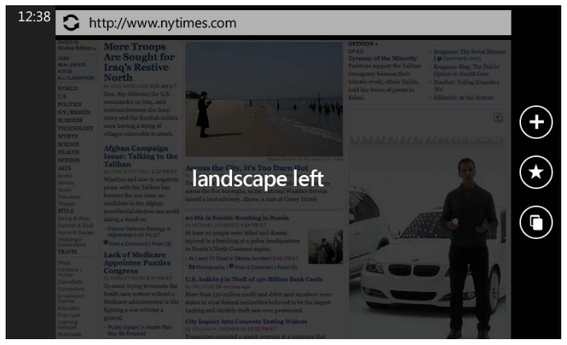
If no menu items are displayed, only the icon text hints are displayed.

The Application Bar Menu will remain on the screen until the user performs an action.

Screen orientations







Windows Phone supports three views of screen orientation: portrait, landscape left, and landscape right.

In portrait view, the page is vertically oriented with the steering buttons appearing at the bottom of the phone and the height of the page is greater than the width.

In either of the two landscape views, the Status Bar and Application Bar remain on the side of the screen that has the Power and Start Button, respectively. Landscape left has the Status Bar on the left and landscape right has the Status Bar on the right.

The Status Bar grows from 32 pixel in portrait view to 72 pixel in both landscape views, as measured from the side of the phone that has the power button toward the center of the screen.

Portrait view is the default view for applications.

Start is always presented in portrait view.

The screen orientation will change based on the following actions:

Beginning Screen Orientation	Rotating	Ending Screen Orientation
Portrait	60 degrees left	Landscape Left
Portrait	60 degrees right	Landscape Right
Landscape Left	60 degrees right	Portrait
Landscape Right	60 degrees left	Portrait
Landscape Left or Right, flat on a table	30 degrees up	Portrait

If in portrait view, the screen orientation will change to either of the landscape views when a user slides out a horizontal hardware keyboard.

There is no programmatic way to switch orientations as the orientation property is set to read-only but it is possible to set a fixed orientation.

Screen transition animation effects are played when screen rotation occurs.

In-application landscape view-aware system components are the Status Bar, Application Bar, Application Bar Menu, Volume/Ring/Vibrate Display, Push Notifications, and Dialogs.

Developers must add code to support landscape views.

Applications cannot specify only left or only right landscape views if they support orientation changes – both views must be supported.

Applications can define a static orientation view using the Supported Orientations property.

Applications that support text input should assume a horizontal hardware keyboard is present and support landscape views.

Custom screen transition animation effects are prohibited.

Fonts

Segoe WP Regular

abcdefghijklmnopqrstuvwxyz1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Segoe WP Bold

abcdefghijklmnopqrstuvwxyz1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Segoe WP Semi-bold

abcdefghijklmnopqrstuvwxyz1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Segoe WP Semi-light

abcdefghijklmnopqrstuvwxyz1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Segoe WP Black

abcdefghijklmnopqrstuvwxyz1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ

The Metro design principles center on a look that uses type prominently throughout Windows Phone 7. Segoe WP is the system font and it is a Unicode font. It has kern pairing, but does not have font hinting. It is available in five styles:

1) Regular
2) Bold
3) Semi-bold
4) Semi-light
5) Black

A standard set of East Asian reading fonts that support Chinese standard, Japanese, and Korean is also included.

Developers can embed their own fonts for use within their application, but they will only be available for use within that application.

Do not post Segoe WP fonts for redistribution or package with an application – this would violate the license terms of the font.

Since Segoe is such an integral part of the UI experience, use alternative fonts sparingly in applications.

Avoid using font sizes that are smaller than 15 points in size. Text that is smaller than 15 points in size can be hard to read and are likely too small in size as touch targets without touch target padding.

If using colorized fonts, use high-contrast colors at smaller point sizes to enhance readability and test against both themes and all accent colors.

Incoming phone calls

When a user receives or places a phone call, the UI of the application currently in view is completely obscured to display the dial pad or information about the incoming call.

Once the call is connected or accepted, the call information flips to the top of the screen and the application appears beneath it in a dimmed-out view. Tapping in the dimmed-out area or pushing a hardware button minimizes the call progress information into a 64 pixel bar in the portrait mode and 75 pixels in the landscape mode. Tapping in the dimmed area brings the obscured application to the foreground for user interaction.

If the keypad or additional call features are selected during a call, the application currently in view is completely obscured.

Call progress information stays pinned to the same side of the phone as the power button and the text does not rotate from the portrait orientation.

If the proximity sensor senses an object near it, it will power off the screen to conserve battery. This happens when the phone is held to the ear or may also happen if the phone is being held horizontally and a finger obscures the sensor. The position of the proximity sensor will vary by phone manufacturer.

Applications that expect user interaction during a phone call should have a minimum 75 pixel margin on the edge of the device that has the power button. No touchable UI elements should be placed within that margin.

Push notifications

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 tile, toast or raw notification.

There are three methods to display push notifications:

1. Tile notifications – Awareness notifications inform users of changes or events that may have occurred and are non-disruptive to the user workflow. They appear in Start tiles. See Application Tiles and Tile Notifications for more information.

2. Toast notifications – 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 user receives a text message or instant message. These notifications appear at the top of the screen and are displayed for 10 seconds before disappearing. See Toast Notifications for more information.

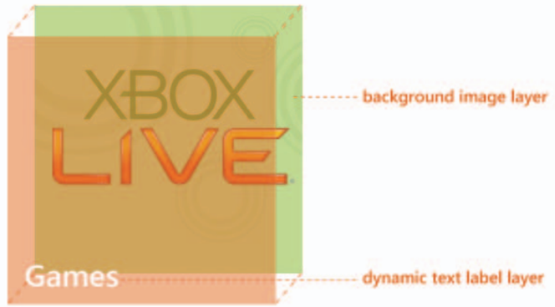
3. Raw notifications – Action-required in-application notifications are fully controlled by an application and affect only that application. These appear within an application. See Raw Notifications for more information.

Use tile notifications for awareness-only notifications.

Use toast notifications for action-requested notifications, but use them sparingly, as all applications have access to toast notifications. Too many toast notifications could annoy or frustrate the user.

Use raw notifications for in-application action-required notifications.

Tiles and tile notification



A tile is an easily recognizable visual shortcut for an application or its content that users can set in an arbitrary location on the phone Start experience. Other than pre-installed application tiles, only the user can pin tiles to Start. There is no method for an application to determine if its tile has been pinned to Start, so developers should not assume that it is.

Tiles can communicate information to the user by displaying an optional counter that uses the system font, updating developer-provided tile background images, or displaying an optional title that uses the system font that is of a fixed size and color. Counter, background image and title updates are controlled using the Tile Notification service. The accent color for the counter is always the accent color that the user has selected. Counter display is optional.

Double-width tiles are only available to Microsoft, phone manufacturers, and mobile operators.

Applications that do not incorporate a tile image or title will display a generic, system-defined icon and the name of your project.

Tile images should be 173 pixels by 173 pixels at 256 dpi and in JPEG or PNG formats. Images larger or smaller than this in size will be cropped or scaled up using the top left corner as the origin. The default tile image will be scaled down for display in the application list unless a separate 63 pixels by 63 pixels application image is included.

The tile title can be displayed without using Tile Notifications.

If you use multiple tile images, they should be visually consistent with each other and have a recognizable theme or style.

Developers cannot change the color, font, font color, or size of the counter display.

Be conservative in the use of Tile Notifications – excessive use can negatively impact battery life.

Toast notifications

A web service can generate a special kind of push notification known as a toast notification, which displays as an opaque bar in the Accent Color on the top of the screen for 10 seconds to be tapped on before disappearing. If the notification is tapped, the application that sent the notification will launch. The toast notification displays a scaled-down version of the application icon in the left corner and two fields of text are available, one bolded title and one normal sub-title. Text that is longer than the display area will be truncated.

Examples would be notifications produced via an instant messaging client or a peer-to-peer oriented application. Turn-based games should use the XNA Framework GamerServices for notifications.

Be very conservative in the frequency and number of toast notifications an application generates. As all applications can access this notification channel, imagine every application on a user's phone sending a toast notification every time an event happened in the application – many people might find this behavior to be very annoying and visually distracting. Follow the guidance closely to prevent user notification overload.

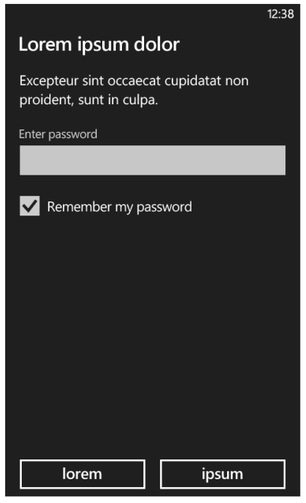
Applications must default to turn toast notifications off.

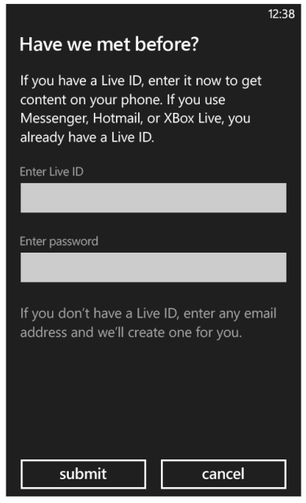
Toast notifications should be personally relevant and time critical to the user.

Toast notifications should primarily be focused on peer-to-peer communication.

Use the XNA Framework GamerServices for turn-based or in-game notifications.

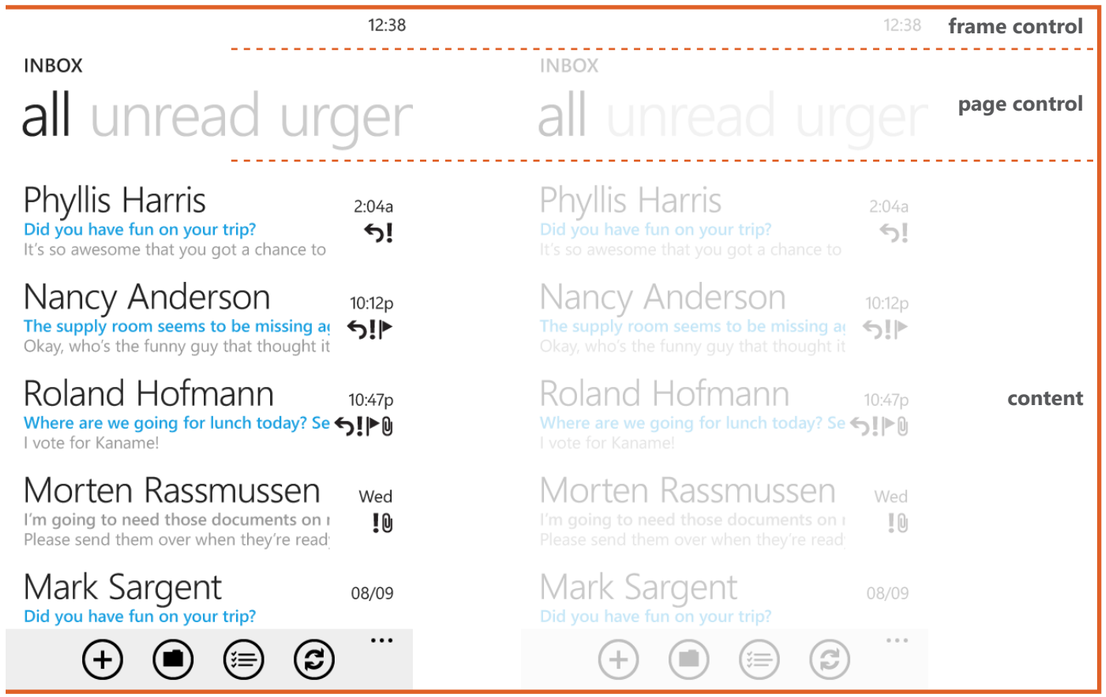
Raw notifications





Raw notifications are in-application, action-requested notifications. They can be generated by the application itself or sent from a web service. Web service raw notifications only appear within the specified application; there is no system-wide way to display a raw notification.

Navigation, frames and pages



Windows Phone 7 applications are based on a Silverlight page model where users can navigate forward through different pages (screens) of content via links and backward using the Back Button. A goal of this model is to ease the creation of view-based applications that fit naturally into the Windows Phone 7 page navigation model.

The core elements of an application include a top-level container control called a frame that displays pages. Only one frame is allowed per application, but there is no limit to the number of pages. Windows Phone 7 provides frame and page classes to facilitate navigation to separate sections of content.

Pages hold discrete sections of content in applications and appear as separate screens to the user. Developers can create as many different pages and construct their UIs as needed to present content within an application and then provide navigation to those pages from the frame or page if desired. Simple applications may only require one page while more complex ones may require many.

Developers can also implement a full-screen view where the Status Bar or Application Bar can optionally be displayed, but this must be explicitly defined using the visibility property, as the default is to not display them. The best practice for a full-screen view is to not display either so that users can focus on the content experience. Notifications and incoming calls are still displayed in full-screen mode, even if the Status Bar and/or the Application bar is hidden. Examples of full-screen UI implementations are a screen animation that is embedded within an application.

The page navigation model is a spoke and hub system. This means that unless developers explicitly add links to other pages within their application, users must use the Back Button to navigate to a page that they wish to view and that users always move forward through the pages. This is similar to how a web browser displays and navigates web page history.

The system tracks each page a user has visited and places it in what is called the back stack so that when a user pushes the Back Button, they are served the last saved page in the back stack. There is no limit to the number of pages that can be placed in the back stack.

The back stack, combined with the hub and spoke model of page

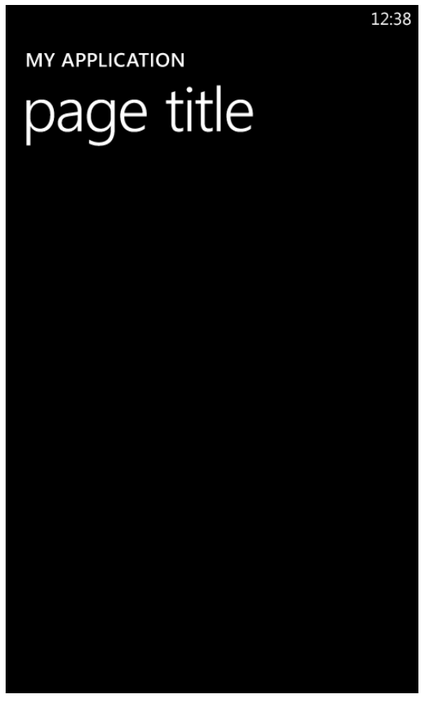
Finding the right number of pages for an application and defining the navigational map may take some trial and error. Mockup the pages and navigational map of an application and walk through them several times before coding to minimize or eliminate the need to add pages or change the map later, when it will be much harder.

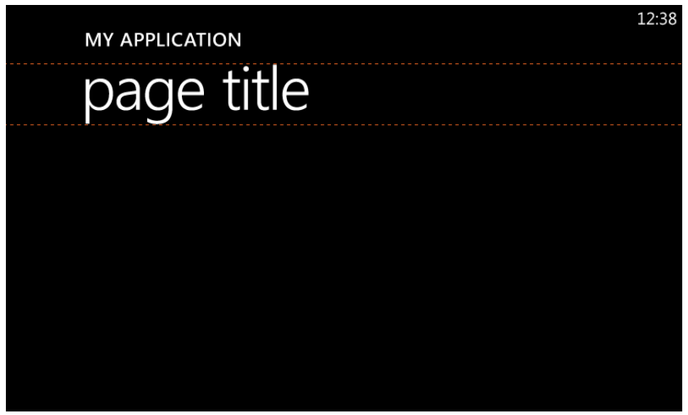
Review the Windows Phone Application Interface Controls section to consider how your application content will fit or be displayed before creating your own custom control for a page.

Do not display the Status Bar or Application Bar when in full-screen mode.

navigation, means that a user navigating from page 1 (p1) to page 2 (p2) to p1 to p2 to page 3 (p3) to p1 creates a back stack of p1, p2, p1, p2, p3, p1. If the user modified content in the second instance of p2 in the back stack, but navigates back using the Back Button to the first instance of p2 in the back stack, unless the page refreshes the data, previous changes will not appear on that page, as it is a snapshot of how the user saw that page at that point in navigation. For this reason, think carefully about implementing page-to-page links or buttons that could impact application navigation for the user and consider if a page should be refreshed upon entry.

Page title





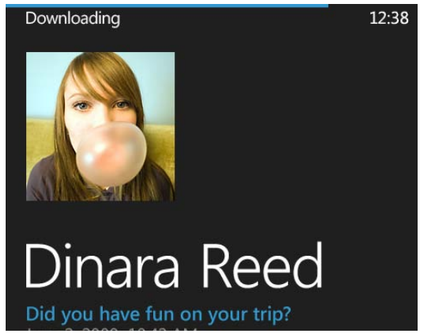
Although not an interactive control, the Page Title is used to clearly display information for page contents. It appears in default Windows Phone Developer Tools templates and is optional. Page titles are not actionable.

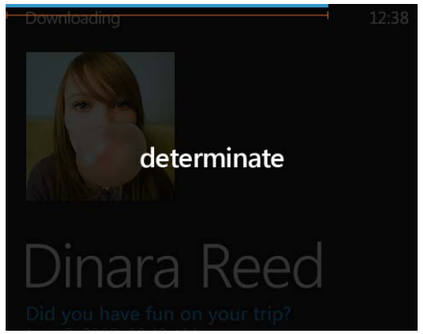
Page Titles are optional. When displayed, they do not scroll.

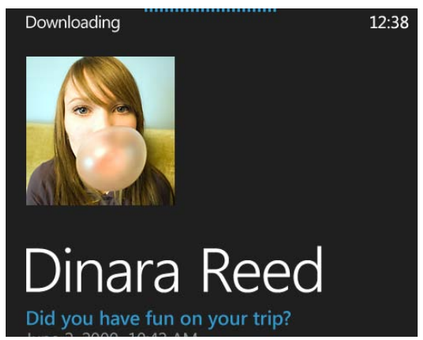
If Page Titles are displayed, reserve the Page Title space in all pages of the application for consistency so the user does not experience differing window sizes across the application.

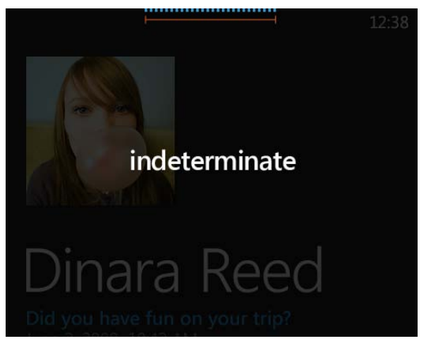
If Page Titles are displayed, the title should be the name of the application or a specific descriptive line of text relevant to the displayed data.

Progress indicator









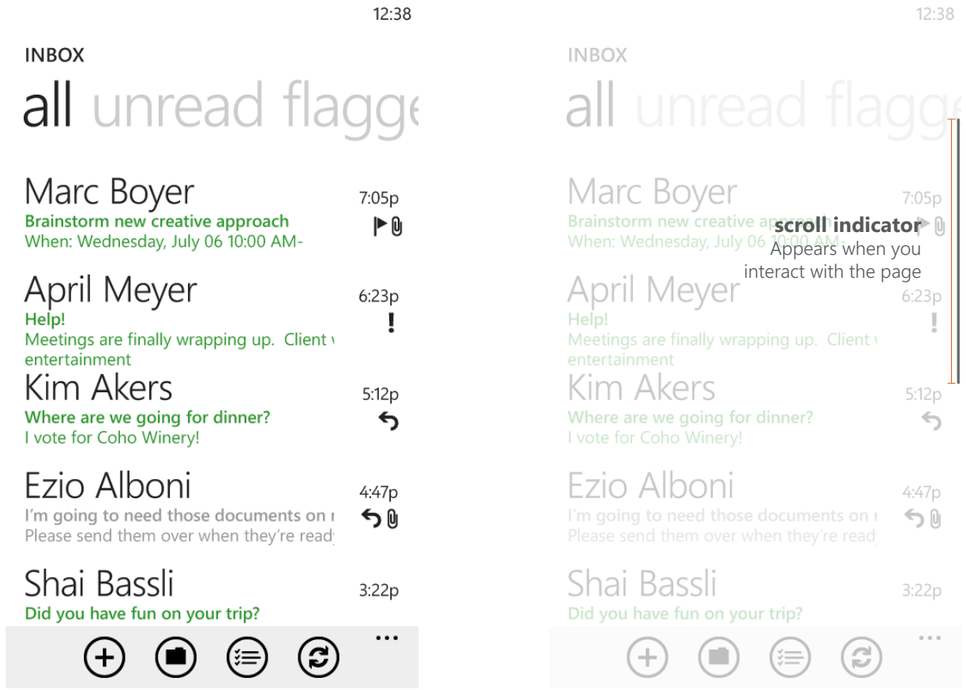
The Progress Indicator shows in-application activity related to an activity or a series of events. This is a system-reserved control that is integrated into the Status Bar and that can be displayed across multiple application pages.

The progress indicator can be either determinate or indeterminate. Determinate progress indicators have a beginning and ending point. Indeterminate progress indicators continue until a task is finished.

See also the Progress Bar topic in the Windows Phone Application Interface Controls section.

Developers who wish to mimic this control should use the determinate indicator for tasks such as downloading content and the indeterminate for tasks such as remote connections.

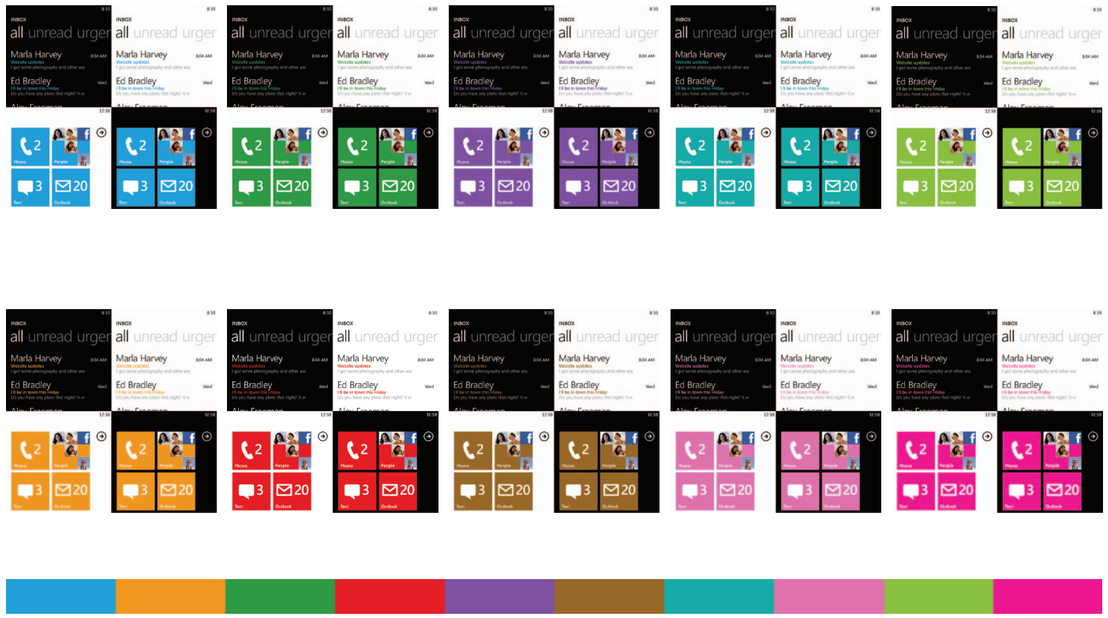
Scroller



Page scrolling occurs when content on the screen exceeds the bounds of the visible page and a user pans or flicks. When scrolling, visible scroll indicators appear on the right side for vertical scrolling and along the bottom for horizontal scrolling to indicate whether the content is longer or wider than the page, and to represent the current position on the page. After page scrolling ends, the scroll indicators fade from view after one second has elapsed.

The scroll indicators are not user actionable and are an overlay to the content beneath. Their primary function is to provide a hint to the user about the page size.

METRO COLORS



A Theme is a user-selected combination of background and accent colors that personalizes the visual elements on a Windows Phone for that user. Only colors are part of a theme. Other elements such as fonts or control sizing do not change.

There are two background colors, dark or light, and 10 accent colors, magenta (FF0097), purple (A200FF), teal (00ABA9), lime (8CBF26), brown (996600), pink (FF0097), orange (F09609), blue (1BA1E2), red (E51400) and green (339933). Mobile operators or phone manufacturers may add one additional system color. The default Theme is a dark background with the blue accent color, but mobile operators or phone manufactures can override this setting.

As a part of the Windows Phone Application Platform, applications automatically take on the selected theme and ensure that system controls and UI elements appear consistently across the platform to prevent a jarring, unsettled user experience.

Developers do not have to adjust application controls to match the user Theme, as these styles will be modified at runtime, but developers can override the Theme within an application. For example, developers may want to override the Theme of an application if they want to build an application that matches a brand color or content consumed from a web service. Developers can provide their own resources and override any themed properties, but cannot turn off theming. Developers should be cautious about using too much white in their applications, as this may have an impact on battery life for devices that have organic LED displays.

Since users can choose between 20 Themes (22 if the mobile operator or phone manufacturer adds an accent color,) developers should consider the possible color combinations if they add colored elements to their UI. Developers may wish to consult with graphic designers for color combination assistance.

Avoid using too much white in applications, such as white backgrounds, as this may have an impact on battery life for devices that have organic LED displays.

User-selected, system-wide Theming cannot be modified; only Themes within applications can be modified.

If the foreground or background color of a control is explicitly set, verify that the content is visible in both dark and light themes. If the set color is not visible, also explicitly set the background or foreground color to maintain contrast or choose a more appropriate color.

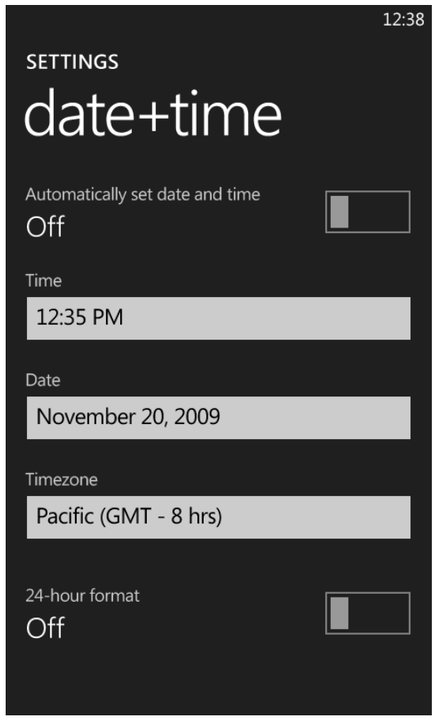
Screen transitions and animations

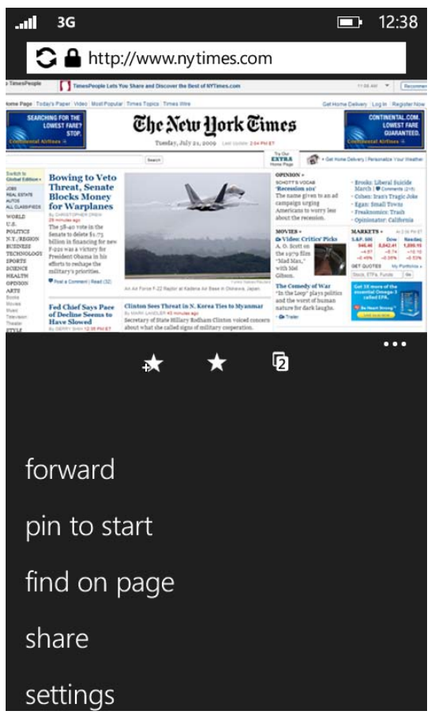
Windows Phone OS 7.0 has many built-in screen transitions and animations that create a sense of a “fluid” user interface. One example is the application entry transition, where unrelated application tiles “flip” out of the way, leaving the selected application tile alone for a moment before it too “flips” to reveal the application user interface.

Built-in screen transitions and animations are system-reserved and developers cannot access them but may mimic them.

If developers want to implement transitions or animations within their application, they must use Silverlight or XNA Framework to create them.

System and system application settings





System and System Application Settings are accessed via the App List and tapping on the Settings icon. Users are presented with a Pivot to view settings choices for the system and for applications that ship with the system. From here, users can personalize the appearance and behavior of their phone by performing activities such as setting the system Theme, joining Wi-Fi networks, or changing the region and language used by the phone.

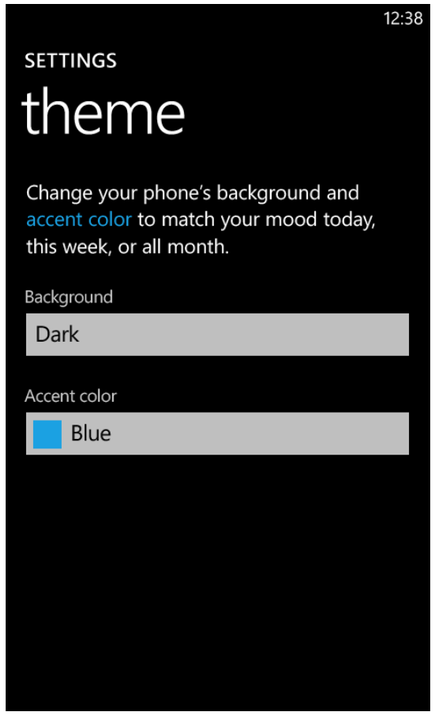
Changes to System and System Application Settings are immediately implemented. In some cases, even though the change has happened immediately, the user may not have feedback that the change has occurred until an ongoing event is completed or when a future event occurs. Examples would be joining a secure Wi-Fi network or changing the frequency of alarms.

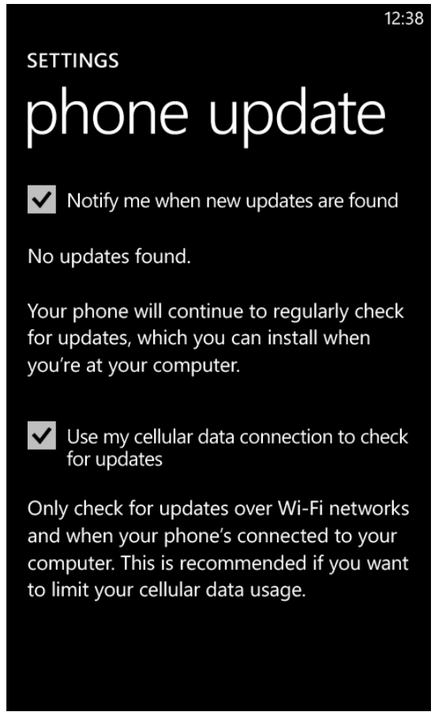
Developers do not have access to place application settings within the System and System Application Settings and must implement application settings pages within the application itself. See the Application Settings topic for more information.

Developers should become familiar with the system settings options and consider how various user settings could impact UI or application behavior. For example, developers of web service-connected applications should consider application behavior when the user puts the phone in airplane mode.

Application settings must be implemented within the application itself.

Application settings





For applications that have several user-selectable settings, developers should create a settings page within the application and model it after the layout and behaviors in System and System Application Settings.

Changes to Application Settings should be immediately implemented. This means that a "Done", "OK", or other confirming dialog is not needed. In some cases, even though the change has happened immediately, the user may not have feedback that the change has occurred until an ongoing event is completed or a future event occurs. Examples would be joining a secure Wi-Fi network or changing the frequency of alarms.

Keeping Application Settings brief and clear should be a design goal. Complex, multi-page, multi-level Application Settings can frustrate or confuse users into thinking that they have entered another application entirely.

Immediately implement user-selected Application Settings without a confirming dialog box and provide a feedback method to indicate that the change has occurred.

Avoid creating Application Settings that have more than 2 pages (screens).

Settings that require more than a single screen should use overlying half screens to avoid losing context when the SIP Keyboard is displayed.

If a task cannot be undone, always provide the user with an option to cancel. Text entry is an example.

Actions that overwrite or delete data, or are irreversable must have a "Cancel" button.

When using additional screens with commit and cancel buttons, clicking those buttons should perform the associated action and return the user to the main settings screen.

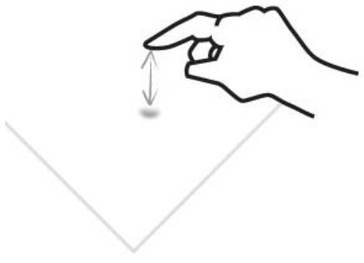
To keep the heading of settings control panels consistent, the heading for the settings page should look as follows:

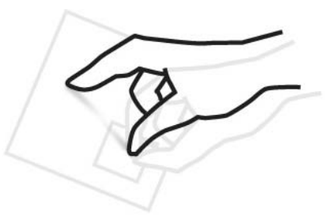
SETTINGS

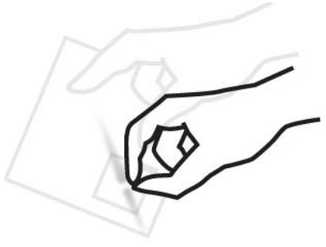
<CPL Name/ Application Name>

Applications that fetch data over the network must have an option to disable data usage.

Input methods







Windows Phone 7 applications can support multiple methods of input:

- Touch
- On-screen keyboard
- Hardware keyboard
- Microphone
- Phone hardware buttons
- Sensors

While not every feature of every input method is available to developers since some are system-reserved, developers should consider each area for applicability to the UI for their applications.

Additional topics in this guide provide greater detail for each input method.

Touch input

Touch input is a core experience of Windows Phone 7 and has inherent differences from traditional keyboard and mouse input systems. Designed for natural and intuitive user interaction, touch input in Windows Phone 7 enables users to interact with application content such as a photo or a web page. Touch input enables simple and consistent user touch gestures that imitate real life behavior, such as panning on a photo to move it. Single-touch gestures make interaction easier with one hand, but multi-touch gestures are also available to provide more advanced gesture functionality.

Application developers should strive to create unique and exciting experiences that encourage the discovery of content through the use of touch gestures. Users should enjoy the experience of navigating through the steps of a task as well as the completion of the task itself. Touch gestures should provide a delightful, more colorful, intuitive experience within applications

Touch delights the senses as the user gets to see the interaction match the performance. The touch UI should always have aware and responsive performance, just like how real world objects respond to touch immediately, and applications on Windows Phone 7 should as well, by performing the action in real time and by providing immediate feedback that an event or process is occurring. Users should not have to wait as it breaks their immersion, flow, and concentration, especially as their gestures transition from one to the other. For example, a pan may turn into a flick or a tap can become a double tap, and the user should not be aware that the UI is switching gesture support.

While this topic provides general touch guidance for customizing UI visual elements, Microsoft recommends that developers use standard Metro-inspired touch controls that are available in the Windows Phone Developer Tools and always follow the guidance when creating custom controls. Windows Phone Developer Tool controls have been properly sized for touch interaction based on the guidelines presented in this section. There are cases where touch UI control sizing will or should vary from the guidance, such as with games, depending on the needs of the application.

Do not use gestures as a shortcut to a task, and only use a gesture in a manner as it was intended. See the Supported Touch Gestures topic for gesture definitions.

All basic or common tasks should be completed using a single finger.

Touch controls should respond to touch immediately. A touch control that lags or that seems slow when transitioning will have a negative impact on the user experience.

Provide immediate visual or auditory feedback to indicate interaction with the touch control. All actions should have immediate and obvious consequence by responding while the gesture happens, not afterwards. A bad example would be a user flicking a photo and the movement occurs after the gesture is completed.

For time consuming processes, developers should elegantly provide feedback to indicate that something is happening by using content to indicate progress, or consider using a progress bar or raw notification as a last resort. For example, show more and more of the content as it is being downloaded.

Response to gestures should be consistent across the phone and within an application. Using the touch controls in the Windows Phone Developer Tools will help with maintaining consistency as they have built-in support for the touch gestures discussed in this topic. If developers create custom touch controls, they should respond to gestures in a similar manner.

Touch input

There are three components to the touch UI:

1) Touch target – the area that is defined to accept touch input and is not visible to the user

2) Touch element – the visual indicator of the touch target that is visible to the user

3) Touch control – a touch target that is combined with a touch element that the user touches

Touch targets should not be smaller than 9 mm or 34 pixels square and provide at least 2 mm or 8 pixels between touchable controls. In exceptional cases, controls can be smaller but never more than 7 mm or 26 pixels square. The on-screen keyboard and hyperlinks in Windows Phone® Internet Explorer® are an exception because they have differently sized hit targets.

Touch targets should be larger than 9 mm when:

- It is a frequently touched control
- Touching it could create a severe error or have a destructive consequence
- The user could become frustrated if they cannot touch it
- It is close to the edge of the screen
- It requires sequential or multiple inputs between adjacent touch controls.

For touch and non-touch UI elements that have special sizing or positioning constraints, layouts may need to be adjusted or additional application pages may need to be created to accommodate the minimum touch target sizes.

Windows Phone 7 gestures align with Windows desktop gestures. There is some intentional variability due to the differences in screen size and the Windows desktop support of the mouse. These differences are mostly around editing shortcuts, which can be potentially addressed by using the on-screen keyboard. Applications for Windows Phone 7 should try to align with the gesture experience of the corresponding Windows desktop application.

Gesture extensibility is not supported. Developers can only use the supported gestures and replicate movement as specified.

Every touch control should be able to be comfortably touched with a finger. This involves manipulating size, spacing, location, and visuals to reduce the difficulty in acquiring a target through finger touch.

Touch targets should not be smaller than 9 mm or 34 pixels square and provide at least 2 mm or 8 pixels between touchable controls. In exceptional cases, controls can be smaller but never more than 7 mm or 26 pixels square.

Touch targets should be larger than 9mm when touch controls are frequently touched, create a severe error such as sending an incomplete message, have a destructive consequence such as deleting data, frustrate the user such as navigating to another screen accidentally, are within 3.5 mm of the edge of the screen, or require sequential or multiple inputs between adjacent touch controls.

Touch input

The touch target can be larger than the touch element, but never be smaller than it, and the touch element must never be smaller than 60% of the touch target.

Use oblong controls in vertically constrained UIs, as these shapes are easier to hit. The touch target height can be as small as 7 mm as long as the width is at least 20 mm.

Supported touch gestures

A touch gesture involves performing a movement with single or multiple fingers on a touch screen. The tapping of a UI element such as a push button is an example of a touch gesture. Touch gestures are the primary method for a user to interact with the Windows Phone UI.

The controls provided in the Windows Phone Developer tools are used as touch interaction elements and are properly sized for touch interaction.

The following single and multi-touch gestures are supported in Windows Phone 7:

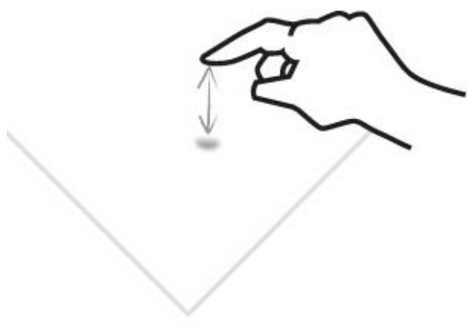
Single-touch:

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

Multi-touch:

- Pinch and Stretch

Tap



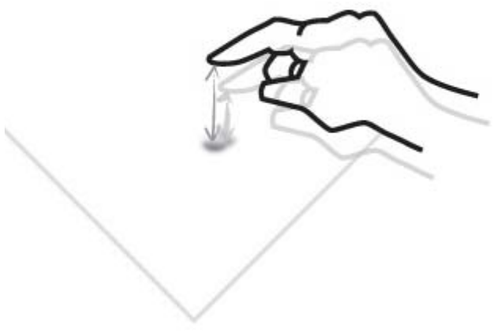
A tap is a single, brief touch on the screen within a bounded area and back up again.

There are two behaviors associated with a tap gesture:

1. Finger down provides touch indication
2. Finger up executes the action

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

Double tap



A double tap is two quick taps within a bounded area.

A double tap toggles between in and out zoom states of a control or an application. The application determines 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

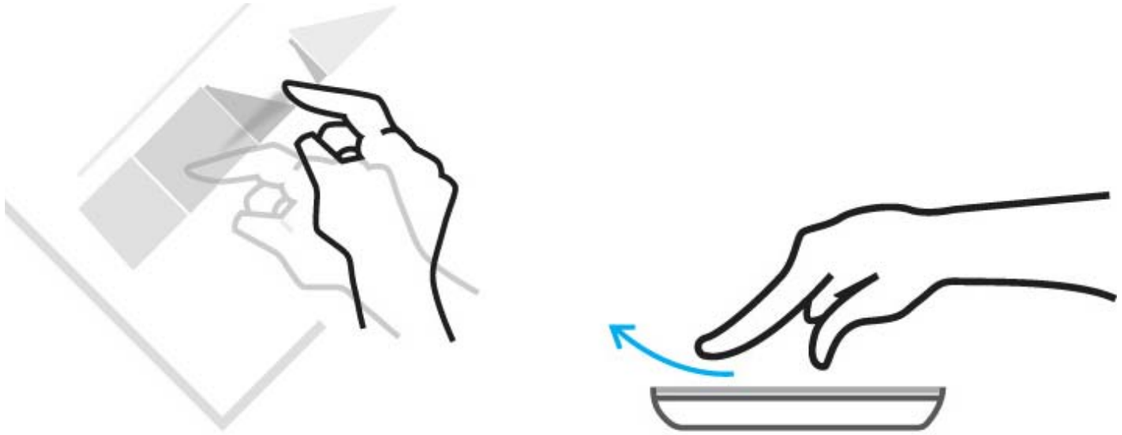
A pan is a single finger placed down and moved across the screen in any direction. The pan gesture ends when the finger is lifted from the screen.

There are two behaviors associated with a pan gesture:

1. Content can be moved through direct manipulation. It will stick to and follow the movement of the finger. Controls or the application can decide what panning direction to support. This movement can be horizontal, vertical, or any other direction specified. If content is moved to an in-between state, the content should snap back to the closest state.

2. 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

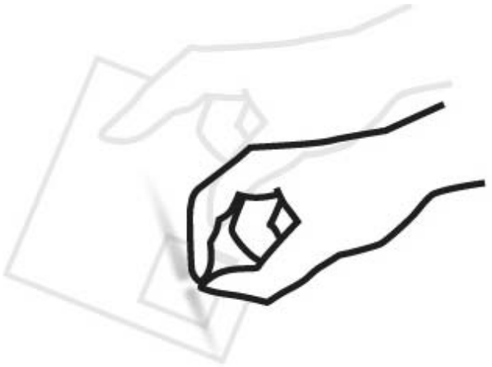


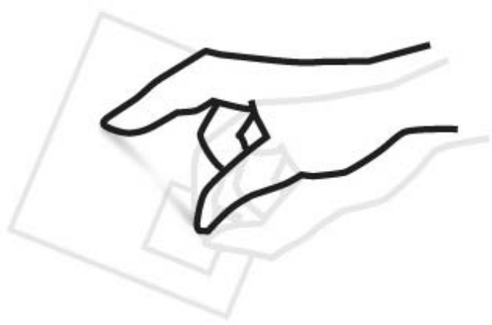
A flick is a single finger down moved rapidly in any direction and ends with the finger up. A flick can follow a pan gesture.

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

Flick moves the whole canvas, but developers can specify individual objects to be moved instead.

Pinch and stretch





A pinch and stretch is two fingers down within separate bounded areas followed by the fingers moving closer together (pinch) or further apart (stretch).

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

Touch and hold

Touch and hold is a single finger down within a bounded area for a defined period of time.

The touch and hold gesture should generally be used to display a context menu or options page for an item.

Four touch points

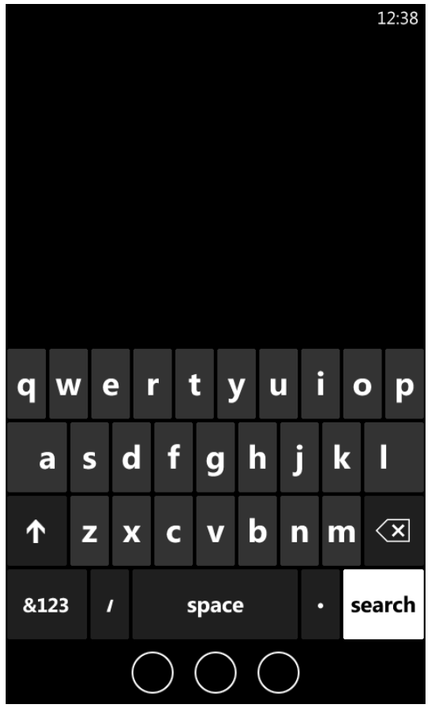
Windows Phone supports four simultaneous user touch input points to enable unique application interactions. Simple examples would be game or musical instrument applications.

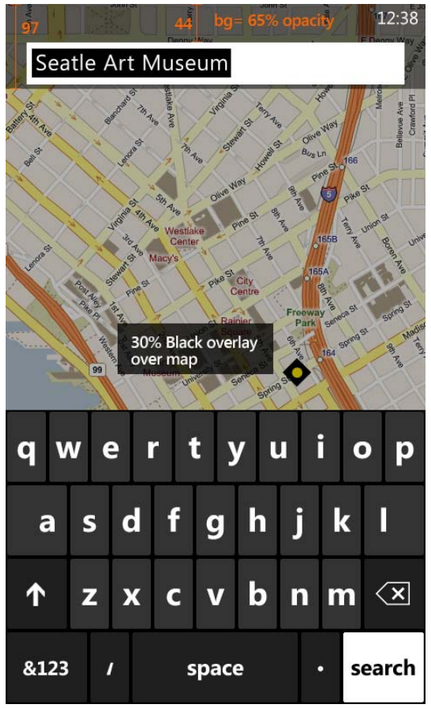
Screen touches 7 mm or larger in diameter are treated as unique touches when the edges of the touch are separated by 3.5mm or more, and all gestures are supported.

Every touch point also adds additional processor load, so developers implementing more than two simultaneous touch input points should be aware of potential performance impacts. While Windows Phone supports up to ten touch points, not all hardware screens will support more than four touch points.

Performance tune applications that support more than two simultaneous touch input points to ensure application performance does not suffer.

On-screen keyboard





The On-Screen Keyboard is for text input and opens by sliding up automatically from the bottom of the screen when an editable control becomes active. When a user taps outside of the edit control, scrolls a list, or presses the Back Button, it closes by sliding down off the bottom of the screen. If a phone has a Hardware Keyboard, (which is a phone manufacturer option,) and is deployed, the On-Screen Keyboard will automatically close.

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

The phone features several typing aids such as text suggestions that appear above the keyboard, auto-correction, and context-specific keyboard layouts.

The On-Screen Keyboard is 336 pixels tall in portrait view and 256 pixels tall in either landscape view. The text suggestion window is 65 pixels tall in both screen orientations.

The developer can define if an edit control is active, and whether or not to deploy the On-Screen Keyboard when a page is navigated to.

When an edit control is active, the system will automatically scroll the editable control above the On-Screen Keyboard if it is a control from the Windows Phone Developer Tools.

If the keyboard has an enter key and the current edit control is a single line, pressing the enter key can either submit the data and close the keyboard or change focus to next edit control. If the edit control is multi-line, pressing the enter key will add a new line.

When the On-Screen Keyboard is deployed in an application, it obscures content behind it.

If text suggestions are enabled, they are docked above the On-Screen Keyboard and obscure content behind it.

If a developer is using a multi-line edit control, part of it may be hidden behind the keyboard. A developer must ensure that the line containing the caret is always in view and above the keyboard.

Developers should set an input scope in edit fields to define the keyboard type 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 must set the input scope property in your project for the text box or edit control.

Only deploy the keyboard automatically if the application page has no more than two edit controls and the first edit control is a single-line edit box. Do not automatically deploy the keyboard if the page has content or controls that would be obscured behind the keyboard.

Do not change the padding inside edit controls or override automatically placed margins of controls places near the edge of the screen. Changing these values can lead to a non-uniform method to select edit controls by touch.

On-screen keyboard

There are 8 different, context-specific, On-Screen Keyboard types that can be used, depending upon the input scope desired:

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

Developers cannot define their own keyboard types or modify existing ones.

The Default, Text, and Maps keyboard types all automatically deploy with the text suggestion window.

If an application takes up all or most of the screen area or has crowded edit controls, it might be difficult for the user to tap outside of the control to close the On-Screen Keyboard and view more content. The application can automatically close the keyboard when the user tries to view the content rather than type, such as when the user scrolls, when the active edit control moves outside of the viewable area, or when the user presses the Back Button. Another solution is to implement an edit view and a read view and open or close the keyboard depending upon the view state.

Hardware keyboard

A Hardware Keyboard is an optional component that phone manufacturers may add. They may come available in designs such as a pull out bar, a vertical slide configuration, or even a flip or swivel orientation.

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

The Hardware Keyboard is only used for typing letters, accented letters, numbers and symbols, and cannot be used to control 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, or 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 not supported or allowed on the Hardware Keyboard:

- A 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 Buttons as part of the keyboard.

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 the accent key is pressed, it adds an accent letter left of the caret. Multiple presses cycle

Hardware keys may not be used to move focus, scroll lists, or navigate maps or web pages.

Hardware keyboard

through the available accents. The function key (FN) plus the accent key cycles back to the previous accent. Pressing and holding the accent key displays 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). Pressing and holding the SYM key displays the language picker. FN and SYM switches to the next language.

A Status Bar input indicator displays shift mode, FN mode, and active language.

Keyboard keys can be overloaded. Pressing and holding a key or using the FN key allows the user to access secondary functionality.

When the symbol picker, accent picker, or language picker is launched, they are displayed at the bottom part of the screen. They disappear after a selection is made or after one second has elapsed.

Applications can use an API to query if the Hardware Keyboard is available or deployed and act accordingly.

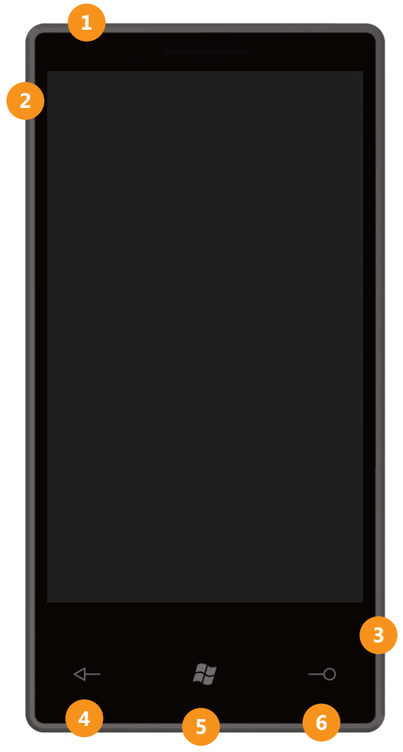
When the Hardware Keyboard is deployed, the On-Screen Keyboard will close. If the device has a fixed hardware keyboard, the On-Screen Keyboard will not be displayed.

Text suggestions are also available for the Hardware Keyboard.

Microphone

The Windows Phone 7 microphone has a frequency range from 150 Hz to 7 kHz.

Phone 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.

1) Power/sleep

2) Volume up and volume down

3) Camera

4) Back

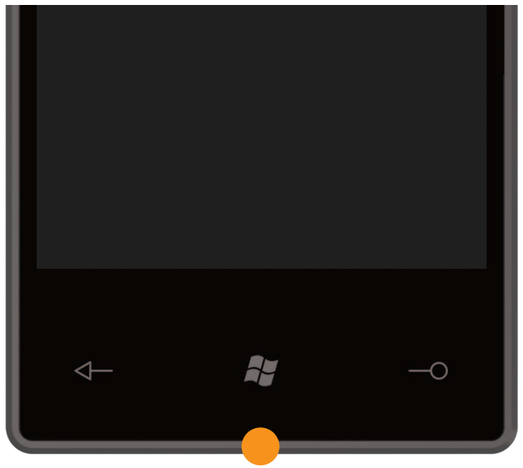
5) Start

6) Search

The Back, Start, and Search buttons can optionally be implemented as capacitive touch buttons by phone manufacturers.

See specific Phone Hardware Button topics for guidance on how each button can affect the UI.

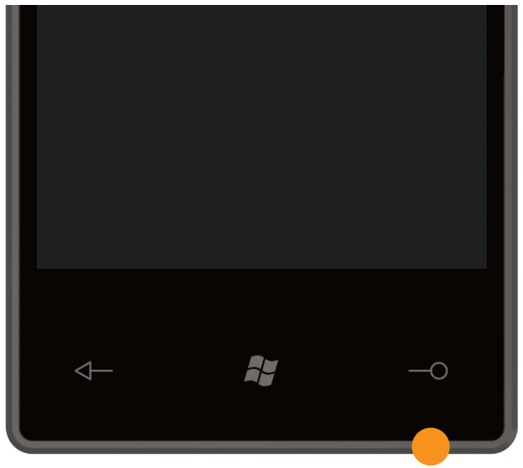
Start button



When a user presses the Start button, it will take them to Start in the phone user interface. Applications will receive an obscured event to pause.

Developers do not have access to modify the Start button behavior and their applications should always be prepared to receive an obscured event to pause after the Start button is pushed.

Search button



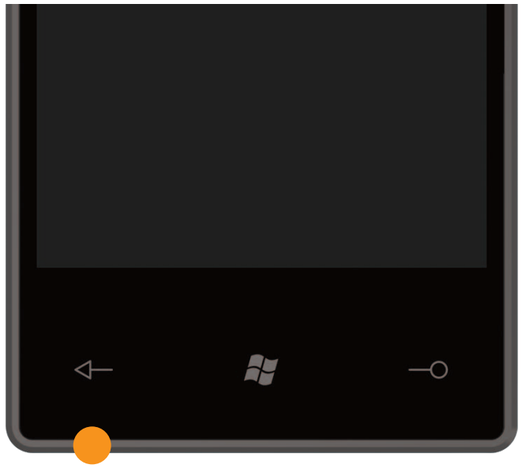
The hardware Search Button will launch the Bing search experience for the user to find content from anywhere on the device. The search experience will vary depending on the context of the user. The Bing search experience can be launched from Start, the App List, system applications, and third-party applications. From select system applications such as Outlook, an in-application search can be launched.

Developers cannot replicate in-application search, but can mimic a Search Button push to launch the Bing search using the SearchTask Class.

Developers cannot modify or change the behavior of the Search Button.

Use the SearchTask Class to launch Bing search from within an application.

Back button



The hardware Back Button is used to navigate back on pages (screens) within an application or between applications. The application allows the framework to perform the operation when the button is pressed. Also, the Back Button can be used to close menus, dialogs, navigate to a previous page, exit a search operation, or even switch applications. However, the principal usage is to navigate from the current page to the previous page.

See the Navigation, Frames, and Pages topic for more information about the page navigation model in Windows Phone 7.

When a user navigates back out of the root page of an application, the application will terminate.

The Back Button will not work as a backspace key to delete text input.

See the Navigation, Frames and Pages topic for additional guidance.

Developers should only implement Back Button behaviors that navigate back or dismiss context menus or modal dialog boxes. All other implementations are prohibited.

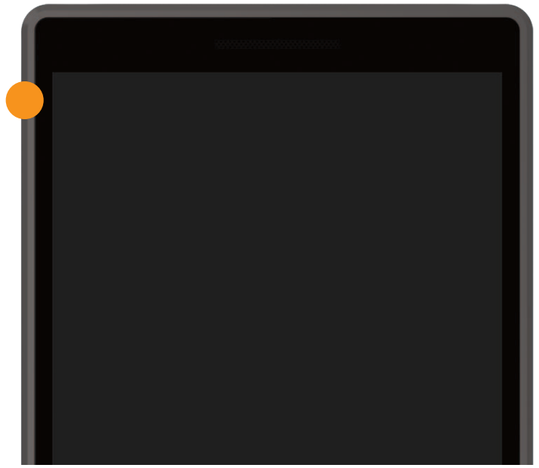
Power button

The power button has several different behaviors depending on the state of the phone:

Developers do not have access to modify the Power button behavior and their applications should always be prepared to receive an obscured event to pause or terminate after the Power button is pushed.

Phone State	Power Button Push Duration	Behavior
Powered off	Any	Boots the phone
Powered on, screen unlocked, screen on	< 3 seconds	Locks and turns off screen
Powered on, screen locked, screen on	< 3 seconds	Turns screen off
Powered on, screen locked, screen off	< 3 seconds	Turns screen on
Any powered on state	> 3 seconds and < 8 seconds	Turns phone off via software shutdown
Any powered on state	> 8 seconds	Turns phone off via hardware shutdown

Volume buttons



The hardware Volume Buttons are used to adjust the volume of either the in-call conversation volume (if a phone call is active), or else the overall device volume (if there is no active phone call), which includes music, radio, video, application, ringtone, and system sound volume.

Pressing either Volume Button will expose the volume control, a 93 pixel tall overlay at the top of the screen and may contain audio transport controls such as previous and next if there is a media player active. It will always contain a control to toggle the ringer setting on and off. This control affects the playback of the system sounds that the user can control in the Ringtone and Sound settings screen.

If the phone is locked, the Volume Buttons are still active when media is playing or there is a phone call in progress.

The buttons operate system-wide and volume settings carry through into the application. This means that developers cannot have volumes set higher than the user settings or override mute.

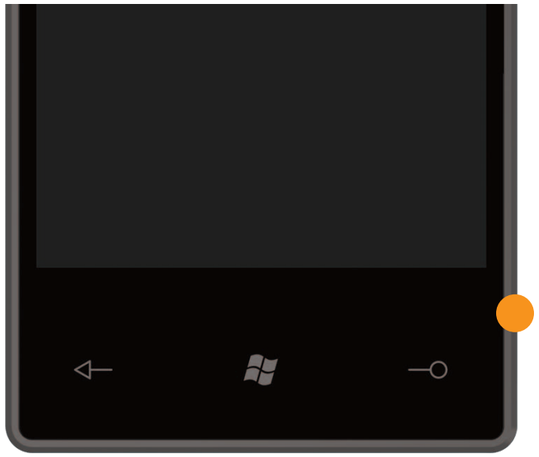
Pressing and holding a Volume Button will do a key press repeat and incrementally increase or decrease the volume depending on the button pushed.

When a user receives a phone call, touching either Volume Button will silence the ringtone.

Developers cannot edit the audio transport controls overlay or override the Volume Buttons behaviors.

Developers can control the volume of the audio stream they provide to the system, including muting it.

Camera button



The Camera Button is a dual action button supporting full and half button press modes. When a user does a full button press, the phone will launch the camera application. If the user does a half button press after the camera application has launched, the auto-focus feature is enabled.

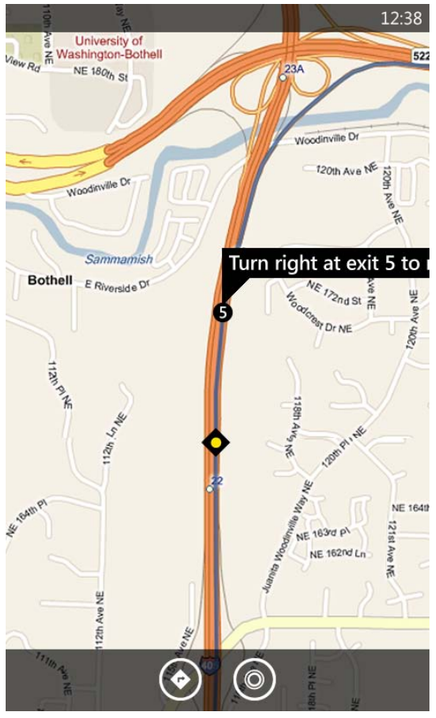
Within the Camera application, pushing the Camera Button will take a photo when in camera mode or start or stop video capture when in video mode.

If the user presses and holds the camera button for more than one second when the device is in stand-by (screen off) or locked, the camera application will launch.

Applications can programmatically launch the camera application by calling the CameraCaptureTask Class.

Developers cannot modify or change the behavior of the Camera Button.

Sensors



Every Windows Phone 7 phone contains the following sensors:

- Accelerometer
- A-GPS
- Proximity Sensor
- Camera
- Compass
- Light Sensor

While not every sensor is available to developers since some are system-reserved, developers should consider each area for applicability to the UI for their applications.

Additional topics in this guide provide greater detail for each sensor.

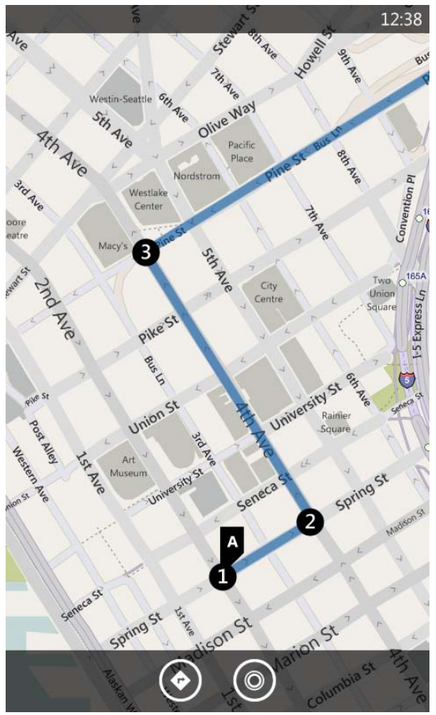
Accelerometer

The Windows Phone 7 accelerometer is an electromechanical device that measures acceleration caused by gravity or external sources to an accuracy of +/- 5 degrees. This 3D motion sensor 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 by calling managed APIs that are easy and flexible to use. Developers can offer scenarios such as automatic screen rotation, tilt-to-scroll, and gaming control within their applications.

Developers creating applications or games that require a higher level of precision and sensitivity from the accelerometer should calibrate accelerometer data.

A-GPS



A-GPS (assisted global positioning system) is used to determine the location of the phone and provides information to Location Services on the phone.

There are no direct UI elements associated with A-GPS, but developers have access to Location Services in the System.Device.Location namespace.

Proximity sensor

The Proximity Sensor is used to power off the screen to conserve battery when an object is detected within 15 mm of the sensor during a phone call. Phone manufacturers have discretion as to where to place the sensor, so its location may vary from phone to phone.

The sensor remains active when the phone is being used in speakerphone mode, so it is possible to accidentally power off the screen if a finger or object covers the sensor.

Developers do not have access to the Proximity Sensor or its behaviors.

Camera

All Windows Phone 7 phones have a five megapixel or larger camera with auto-focus and flash, and a 4:3 aspect ratio image sensor.

There are no direct UI elements associated with the Camera, but developers have access to the Camera in the Microsoft.Phone.Tasks namespace.

Compass

The Compass is used to determine navigation direction relative to the Earth's magnetic field.

Developers do not have access to the Compass or its behaviors.

Light sensor

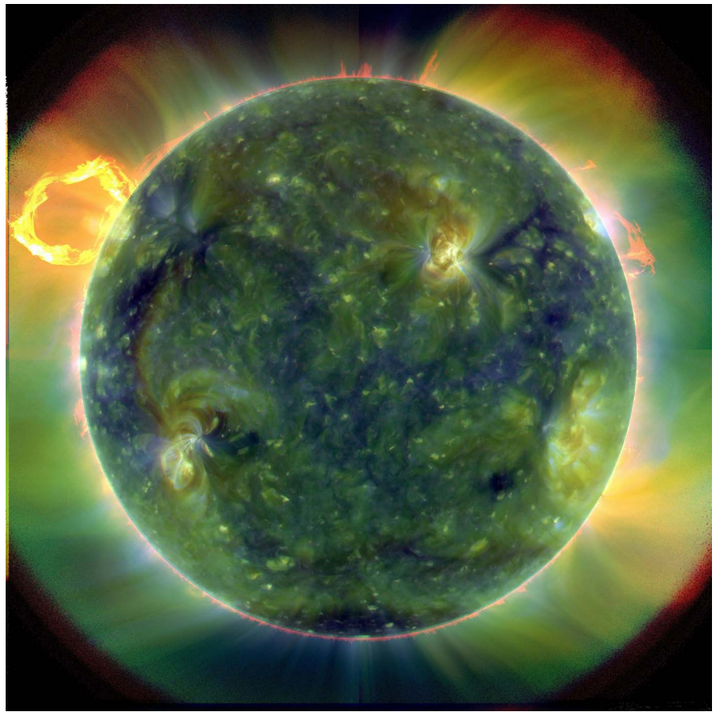
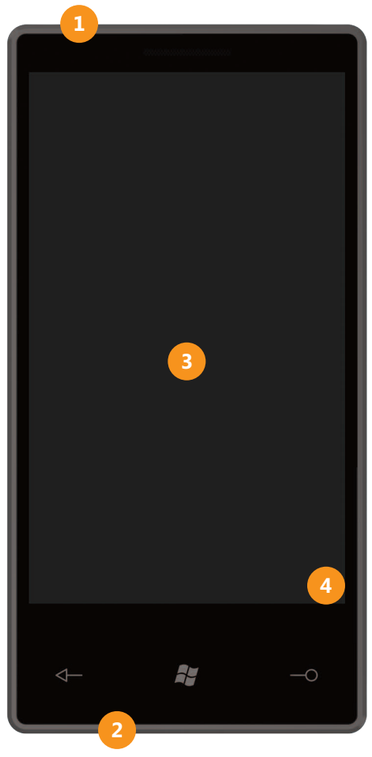


Image Source: NASA

The Light Sensor is used to determine the intensity of light up to 4,000 lux and helps in automatically adjusting screen brightness. Developers do not have access to the Light Sensor or its behaviors and there are no UI elements associated with it.

Output methods



Windows Phone 7 applications have multiple methods of output:

1) Audio output jack

2) On-device speaker

3) On-device screen

4) Vibration

Developers should consider each area for applicability to the UI for their applications.

All Windows Phone 7 phones will have at least 16-bit (5 red, 6 green, 5 blue) WVGA screens at 800 x 480 pixel resolution, no matter the screen size. Phone manufacturers have the option to use higher bit depth screens, but there is no programmatic way to query for bit depth.

The vibration unit can be turned on or off by the user in the Ringtones and Sounds preferences and this setting cannot be overridden.

If developers or designers require precise millimeter sizing for UI elements, consult the display specifications provided by original equipment manufacturers for the proper pixels to millimeters conversion factor. See the topic, A Note on Units – Pixels vs. Millimeters for more information.

Consult the display specifications provided by original equipment manufacturers for screen bit depth.

FM radio

All Windows Phone 7 phones have a built-in FM radio tuner. Developers will need to create the UI for the service, as there are no built-in UI components.

Use the base Silverlight controls in the Windows Phone Developer Tools to build application UIs.

Applications cannot lock the frequency or region of the FM radio tuner; therefore, developers should poll the API prior to displaying the UI to refresh the frequency or region values of the tuner in case another application has changed the values.

Windows phone application interface controls

The Silverlight® UI framework delivered on Windows Phone 7 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 platform. Developers can use their previous Silverlight and .NET development experience to facilitate working with this mobile control set and apply it to their Windows Phone 7 applications.

The following topics list the Silverlight controls in the Windows Phone Developer Tools, including controls that do not have UI components but can affect other parts of the UI. Refer to specific control topics for a detailed overview and guidance for each. This guide only focuses on controls that impact the phone UI and does not exhaustively cover every Silverlight control available in the Windows Phone Developer Tools, as UI guidance exists elsewhere for those controls.

User set system theming flows down automatically into all controls available in the Windows Phone Developer Tools.

Developers may also create their own controls as needed and these controls may replicate system controls that are not available as a part of the Windows Phone Developer Tools. See the Visual Design Resources topic for graphic design templates for system controls that developers may wish to mimic.

If you need to create your own user actionable control, please adhere to the general Metro styling guidance as discussed in the first section of this guide, Philosophy of the UI and Design.

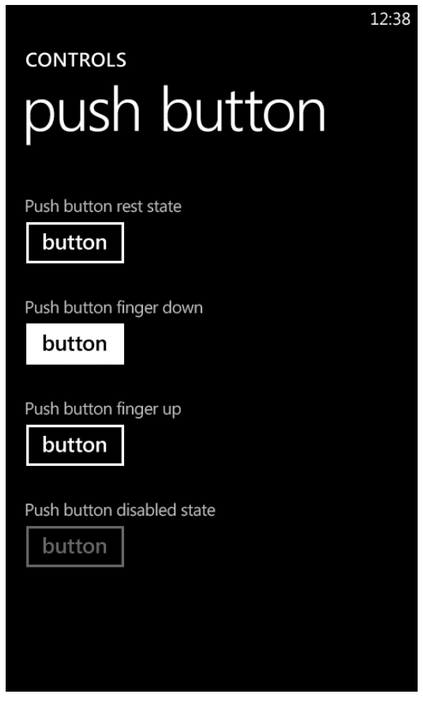
Use the Windows Phone Design Templates as visual guides for creating custom controls that mimic system controls.

Border

Provides a border, background, or both to another control. A border can contain only one child element.

Do not hard-code the border width. Use the phone border width to support screen scaling.

Push button



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

Buttons support rest, press, and disabled states.

There is no visible focus state.

Buttons support the tap gesture.

If using text, buttons should never include more than two words.

Text should be concise and typically a verb.

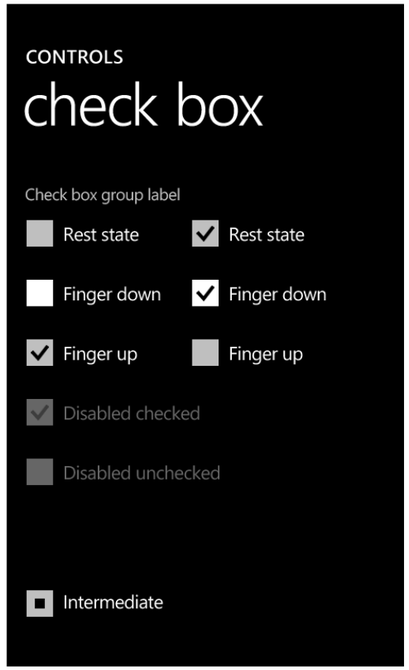
When used in dialog boxes, “OK” or positive actions should be on the left, and “Cancel” or negative actions on the right.

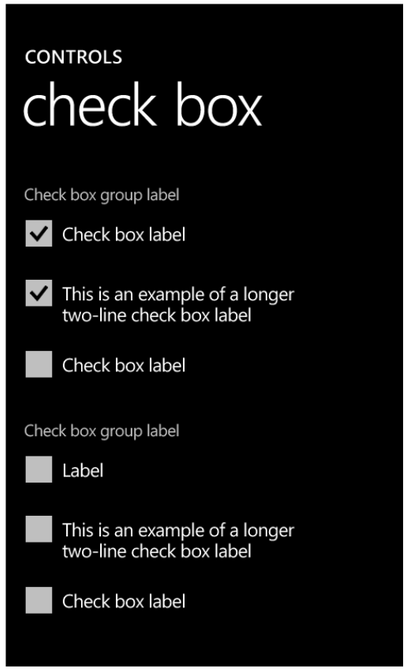
Canvas

Provides a surface to display child elements at specific coordinates in the canvas.

Canvas uses a pixel-based layout and can provide better performance than using the grid control for deeply embedded or nested controls in layout passes for applications that do not change orientations. Use the grid control if the application frame needs to grow, shrink, or rotate.

Check box





Check boxes are used to define a binary state and can be used in groups to display multiple choices from which the user can select one or more choices. A user can either tap a check box or the accompanying text to select an option.

The control supports an indeterminate state that can be used to communicate checked and unchecked status simultaneously for a number of underlying items.

Check boxes support rest, press, and disabled states for both selected and un-selected settings.

There is no visible focus state.

Even though the control can support multiple lines, limit text to either a one or two line format for design consistency.

If there are several choices to present to a user, consider using a scroll viewer and adding a stack panel.

Microsoft does not recommend the use of the indeterminate state because the user could be confused about which of the underlying property items is actually checked or unchecked. A more appropriate alternative is to map the data sources for that checkbox to separate checkboxes or use a multi-selection list, particularly if a dynamic data set is used.

Content control

Represents a control with a single piece of content of any type. Many controls derive from Content Control and can contain objects, such as a Button.

Content presenter

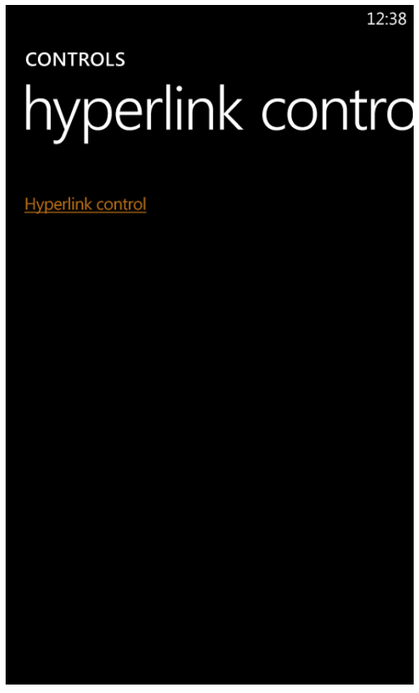
Displays the content of a Content Presenter.

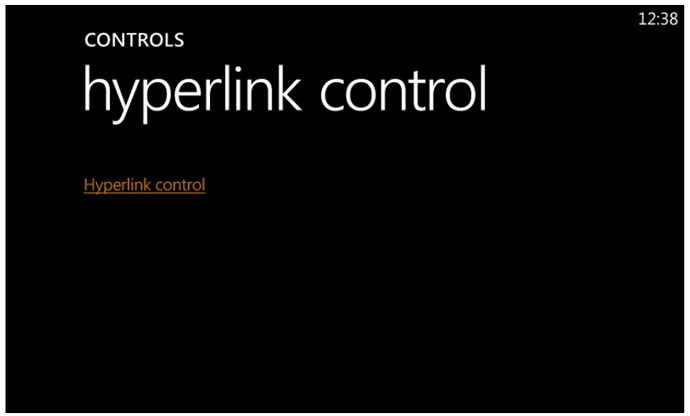
Grid

Provides a surface that is composed of rows and columns to display child elements. You define the rows and columns for a grid, then, assign objects to a specific row or column in the grid. You can optionally display gridlines.

Using the grid control for deeply embedded or nested controls in layout passes for applications that do not change orientations or grow or shrink the viewable area can lead to degraded performance. Use the canvas control if the application frame does not need to grow, shrink, or rotate.

Hyperlink





The hyperlink control allows you to embed hypertext links in a page and specify a navigation target.

The control supports rest, press, and disabled states.

There is no visible focus state.

Only use the Hyperlink control for navigation, not to trigger events, or hide or show additional text. To trigger events, use a button control instead.

Avoid placing hyperlinks close to each other. Doing so may make it difficult for the user to select an individual link without zooming.

Hyperlink 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 disabled and cannot be enabled by user action should not be displayed.

Image

The image control displays an image in PNG or JPEG format and displays indexed images with 1, 4, or 8 bit color-depth or true color images with 24 or 32 bit color-depth.

InkPresenter

InkPresenter provides a primitive drawing surface to collect strokes or Bézier curves within a canvas control.

InkPresenter does not support handwriting recognition.

ListBox

A ListBox control contains a collection of items. You can populate the control by binding it to a data source or displaying unbound items. The ListBox is an items control, which means that you can populate it with items that contain text or other controls.

MediaElement

A MediaElement control provides a rectangular region that can display video on its surface, or plays audio if no video is present.

Do not use this control for sound effects in your application; use the XNA Framework SoundEffect API instead or your application will fail certification. This is because the MediaElement will interrupt and halt any audio that is playing in the background.

Do use this control for full-screen video playback, or in other situations where background audio would be halted.

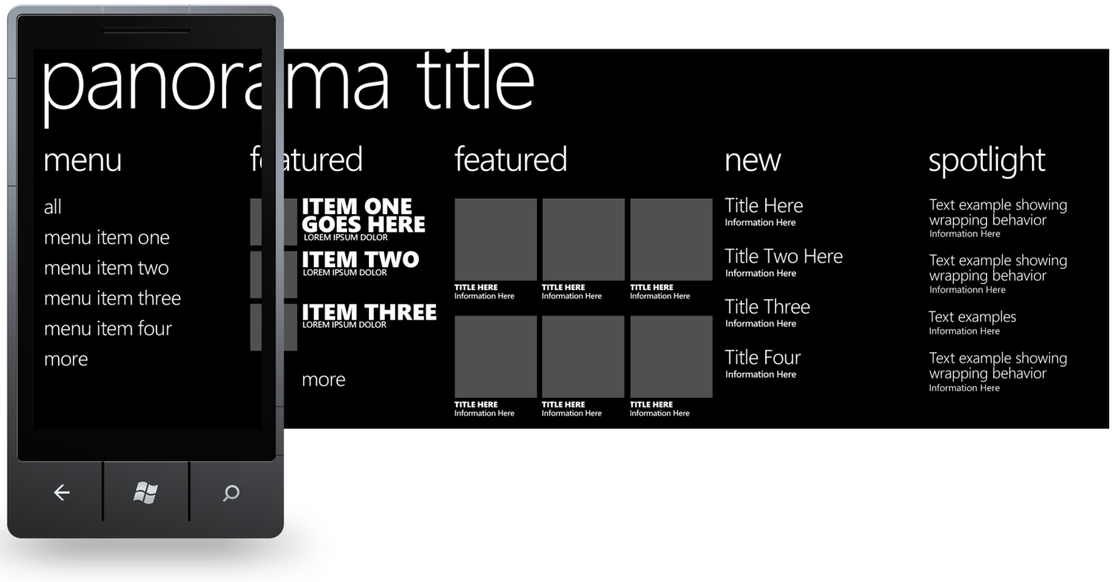
Only one MediaElement can be active at a time.

Multi scale image

Multi Scale Image enables users to open a multi-resolution image that can be scaled and repositioned for detailed viewing. By default, an image that is loaded by Multi Scale Image zooms (expands) when first loaded. This behavior can be disabled by setting the UseSprings property to false.

Developers must implement gestures when using this control because it does not have any gesture support built-in.

Panorama



Panorama applications are a part of the core Windows Phone OS 7.0 visual experience. Unlike standard applications that are designed to fit within the confines of the phone screen, these 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.

Thumbnails are a main element of the panoramic view. They link to content or media that is consumed outside of the panoramic experience.

Elements of a panoramic application serve as the starting point for more detailed experiences. The element flow example is not indicative of platform functionality, but rather the end-user experience. As an example, while an application that is launched from a panorama application might be what the end-user sees; the launched application is actually just a different view of the same panorama application.

The user interface consists of layer types that operate with their own independent motion logic: a background image, a panorama title, panorama section titles, and panorama sections. Thumbnails complete the experience and are a main element of the panoramic view. They link to content or media that is consumed outside of the panoramic experience.

The background image is the lowest layer 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.

The panorama title is the title of the overall panorama application. It is meant to let the user identify the application and should be visible no matter how they enter the application.

Panorama sections are the component of the panoramic application that encapsulates other controls and content. Panorama sections move at the same rate as the finger pan or flick.

A panorama section title is optional for any given panorama section.

Microsoft will provide a control and design template to support building Panoramas at a later date. Microsoft recommends that designers and developers wait to implement Panoramas until final guidance is available and to use the following information for planning purposes only.

For Panoramas:

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 that is supported by Silverlight is acceptable, but JPEGs are recommended, as they generally have smaller file sizes than other formats.

You can use multiple images as a background, but you should note that only one image should be displayed at any given time.

Background images should be between 480 x 800 pixels and 1024 x 800 pixels (width x height) to ensure good performance, minimal load time, and no scaling.

Use a 16 x 9 aspect ratio for a panorama application that has four sections.

Use a transparent black or white filter to aid text legibility.

Use a rate of motion that is relative to the panning gesture, which is determined by the total width of the top content layer relative to the width of the background image.

Wrap off and then back onto the visible area when the user pans beyond the width of the image.

For Panorama Titles:

Use either plain text or images, such as a logo, for the panorama title. You can also use multiple elements, such as a logo and text (or other UI elements).

Ensure that the font or image color for the title works across the entire background and that it is not dependent on the background image for visibility. Use the system fonts and styles unless there is a need for a specific branded experience that uses a different font, size or color.

Use the same panorama title for the launch tile in Start for consistency.

Avoid animating the panorama title or dynamically changing its size.

Use a rate of motion for the panorama title that is slow relative to the topmost content layer, and slower than the background art.

For Panorama Section Titles:

Use plain text for panorama section titles. Alternatively, you can use images. You can use multiple elements, such as an image and text (or other UIElements).

Ensure that the panorama section title is not dependent on the background art.

Avoid animated panorama section titles because the title will be moving.

Panorama

Span the entire panorama section with panorama section titles, even if multiple controls are present.

Animate panorama section titles off the screen when a user navigates to a new section.

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 at 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, the title should move along with the content.

For Panorama Sections:

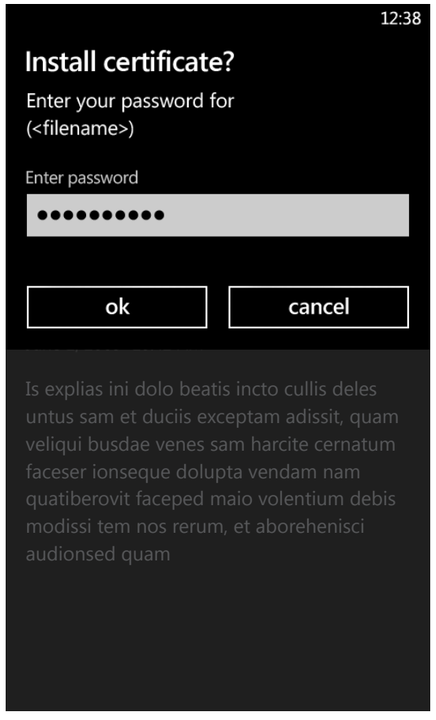
Vertical scrolling through a list or grid in panorama sections is acceptable as long as it is within the confines of the section and is not in parallel with a horizontal scroll.

Animate panorama sections off the screen when a user navigates to a new section.

Consider hiding panorama sections until they have content to display.

For thumbnails, 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 can be used to identify the content.

Password box



The Password Box control displays content and allows the user to type or edit the contents. An entered character appears briefly and is obfuscated to a bullet when the next character is entered or after two seconds.

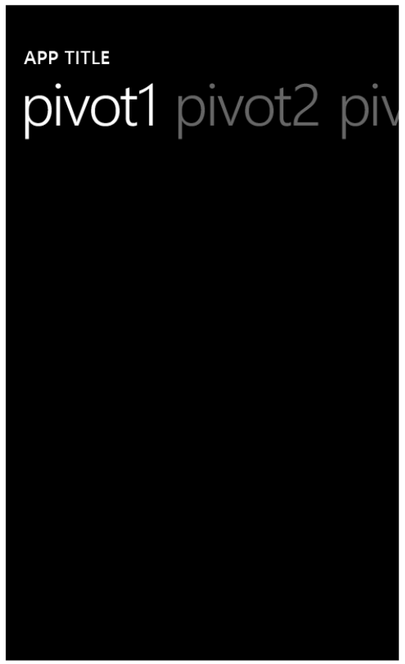
The on-screen keyboard appears automatically when focus is set in the password box unless the phone has a hardware keyboard.

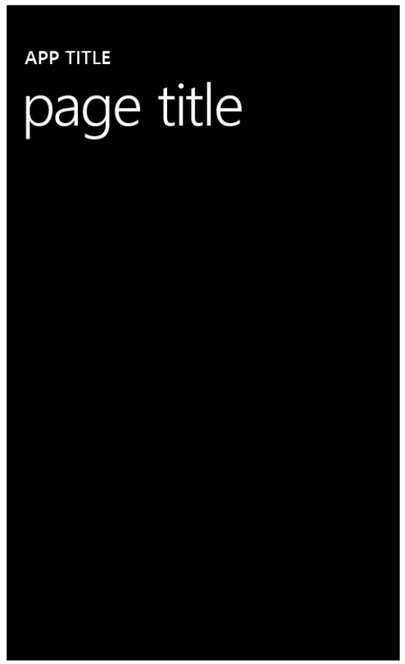
To add an input scope for an on-screen keyboard, configure the input scope property on the password box control.

Gestures supported:

- **Tap – for focus and selection**
- **Tap and hold – for precise caret insertion**

Pivot





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

The content of the page inside the pivot control is defined by the application.

Microsoft will provide a control and design template to support building pivot experiences at a later date. Microsoft recommends that designers and developers wait to implement pivot experiences until final guidance is available and to use the following information for planning purposes only.

Never place a pivot control inside of another pivot control.

Never place a pivot control inside of a panorama control.

Applications should minimize the number of pivot pages. Users can become lost if they jump from pivot page to pivot page. Use pivot controls sparingly and limit the use of pivot pages to scenarios where it is appropriate for the experience.

Pivot pages should not be used for task flow. Different pages should flow seamlessly in terms of look and feel and user activity should not be changed drastically.

Pivot pages must not override the horizontal pan and flick functionality as it collides with the interaction design of the pivot control.

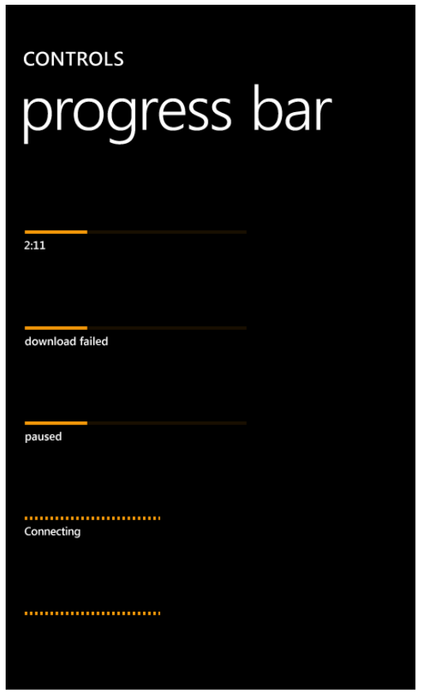
There is no limit on the text length of the pivot header. The amount of text that is displayed is constrained by the width of the pivot control.

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.

An empty pivot page should only be removed when there is no chance that additional information can be added through user action.

Progress bar

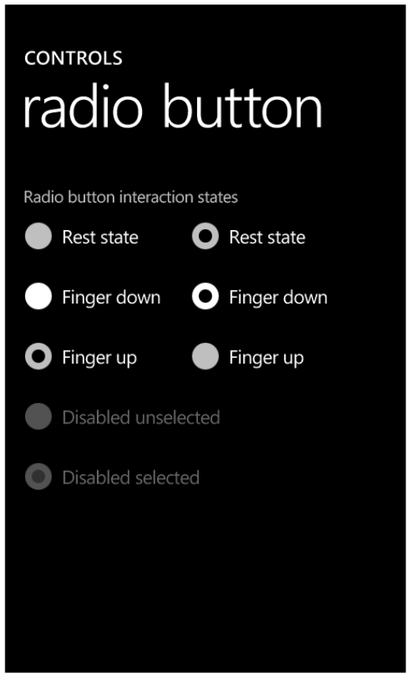


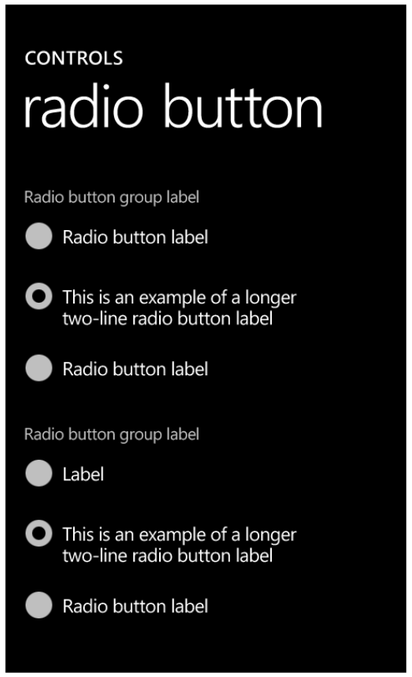
The progress bar is a control that indicates the progress of an operation. You can use the control to show generic progress, or progress that changes according to a value.

It supports a marquee (indeterminate) mode.

The use of a progress bar is optional, but consider adding one if there are wait states in your application that do not require user interaction.

Radio button





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

The radio button control implements rest, press, and disabled states for both selected and un-selected settings.

There is no visible focus state.

The tap gesture switches between selected and un-selected settings.

Radio button description text can wrap to a second line, but try to use either a single or double-line format for design consistency.

Use a scroll panel instead if there are several choices for the user.

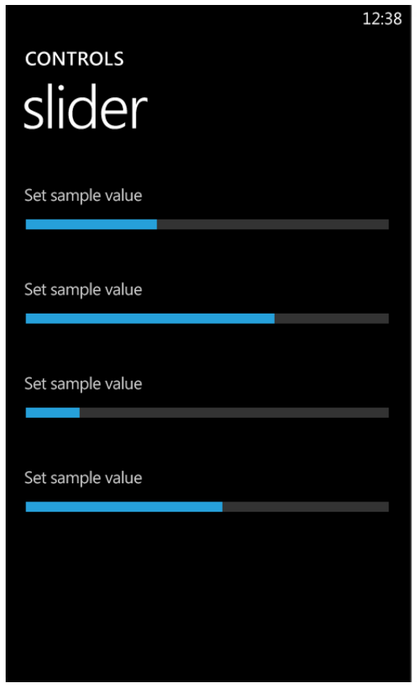
Scroll viewer

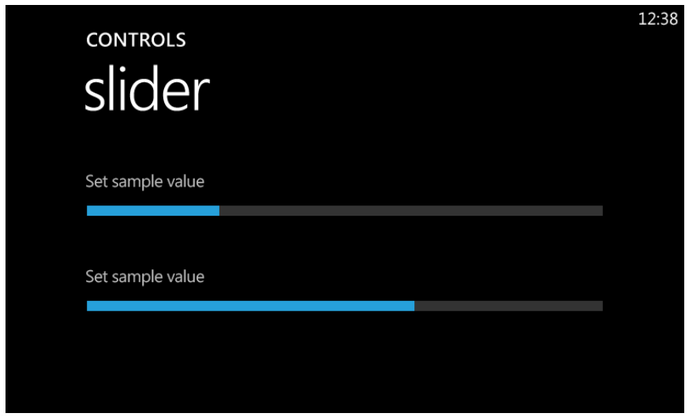
The scroll viewer allows users to navigate to content that is not directly viewable within the frame of the application, such as a long section of text or image.

When scrolling, scroll indicators will fade in as the user pans or flicks and fade out after a second at the end of the gesture, but the scroll indicators are non-user actionable.

The Scroll Viewer supports pan and flick gestures.

Slider





A slider control is used to set a value from a continuous range of data such as volume or brightness levels. The slider has a minimum and a maximum increment value.

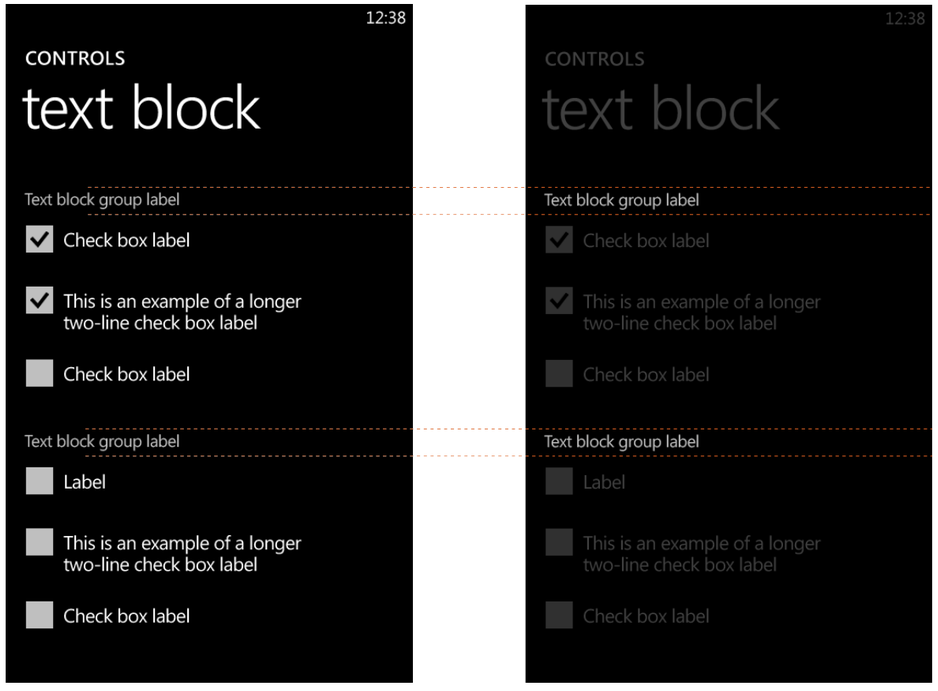
Applications can use either a horizontal or vertical slider, but a horizontal slider is recommended.

Stack panel

Provides a surface to display child elements in a line; either horizontally or vertically.

Padding may be required to prevent crowding or overlapping when placing text that is not part of a control in a StackPanel.

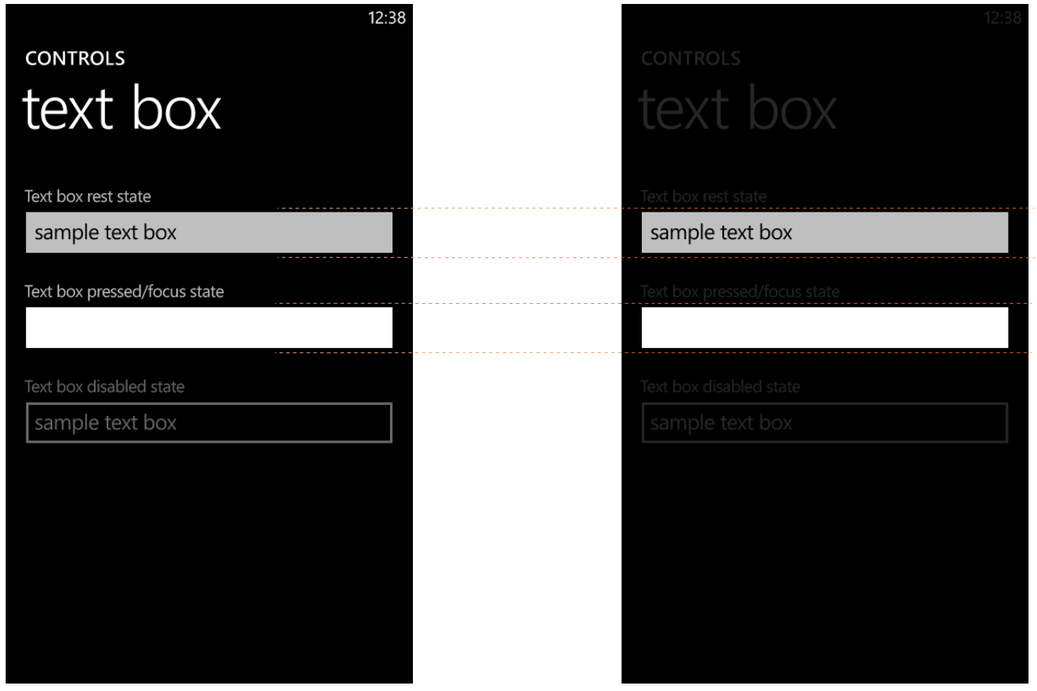
Text block



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 and supports word wrapping.

Always use one of the Windows Phone OS 7.0 predefined text styles instead of hard coding font size, color, weight, or name to support future screen resolutions or sizes.

Text box



The Text Box control displays content and allows the user to type or edit the contents.

Text boxes may display a single or multiple lines. Multiple line text boxes will wrap text to the size of the control.

Text boxes may be set to read only, but should generally be used for editable text.

The on-screen keyboard appears automatically when focus is set in the text box unless the phone has a hardware keyboard.

Gestures supported:

- **Tap – for focus and selection**
- **Tap and hold – for precise caret insertion**

Lorum ipsum

Alit modit eat. Sa doloraes dolor anis maiosam inventorum et aut eium quistius explitiis dit am harchici utent adite comnihilis eiusae ma pa dolorerfere, sitatem hil istione stisquae et illacepudis isto tet molupienet quis sit, tem ea volupta speleni stiusan dentus endae dem quia secessed ene etusa poria si offic tem fugit, nest, tem aut vollaut vendiat volut aute pos vollorest quo od que nam delibea tumquibus none vendestrum harum sit que doloreium quid et occum dolorectinis eiminu stinvende dolest, sere, aliquibus doluptatquia et lacia voluptat aut lamus dolore dolutatem sed et iligent es seditam sum erferit, tecabo.

The Metro design elevates text to a primary UI element and having informed, cleaner, and friendlier text for users will help applications parallel the native Windows Phone 7 text format.

Using the Metro style in voice and tone, terminology, capitalization, and punctuation will add an extra layer of fit and finish to any application and can significantly enhance usability of the application by the user.

Review the Voice and Tone, Terminology, Capitalization, and Punctuation sections for specific guidance in those areas.

Text guidelines – voice and tone

Many users consider text displayed on computers to be another language called computerese, a jargon-filled, soulless, completely impenetrable foreign language that torments them by hindering their ability to complete tasks and asks nonsensical questions in dialog boxes.

Windows Phone 7 banishes computerese entirely and developers should as well. The Windows Phone 7 voice and tone should be human, clear and consistent.

Voice refers to the personality within the text. For example, the voice of the writer would be their overall personality expressed by what they write.

Tone is the overall mood of the text such as happy or angry. The Windows Phone 7 tone is friendly, lighthearted, and empathic.

One method to check if text has the proper voice and tone would be to see if it sounds like a friend assisting another friend with something on the phone. An example would 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 such as, "Error Code: 4B696C626F." Many people could be confused or frustrated by that message, as it provides no context or potential solution. However, something such as, "There is some info missing here. Please enter your name in the text box to move to the next page," is clear, friendly and provides a helpful suggestion.

It is imperative to give users a meaningful response in a casual, comprehensible manner. Help them fix the problem in a way that they can understand.

Consider the string, "Synchronize the phone device." It sounds mechanical and stilted. Instead, "Sync your phone," sounds much more like what someone would tell a friend to do.

Another example is, "Schedule a calendar event for tomorrow through Outlook." It is neither friendly nor representative of how a friend would speak. An alternative could be "Set up an appointment for tomorrow in Outlook."

Do not use computerese – jargon, hexadecimal error codes, or text that assumes computer knowledge.

Use an authentic and clear voice, and reflect the language that is used by the audience.

Use friendly, lighthearted, and empathic tones. Never use an angry or mechanical tone in the application.

If an application has many text strings, consider consulting with a technical writer or editor to review the text strings.

Text guidelines – capitalization

Alit modit eat. Sa doloraes dolor anis maiosam inventorum et aut eium quistius explitiis dit am harchici utent adite comnihilis eiusae ma pa dolorerfere, sitatem hil istione stisquae et illacepudis isto tet molupienet quis sit, tem ea volupta speleni stiusan dentus endae dem quia secessed ene etusa poria si offic tem fugit, nest, tem aut vollaut vendiat volut aute pos vollorest quo od que nam delibea tumquibus none vendestrum harum sit que doloreium quid et occum dolorectinis eiminu stinvende dolest, sere, aliquibus doluptatquia et lacia voluptat aut lamus dolore dolutatem doluptatur arcipidunt adictur sincimos eati audignis nim dolorita coribus

Windows Phone 7 displays text in lowercase and all caps layouts in many places, but also uses title caps, where the first and last words of the phrase and all words in between are capitalized, and sentence caps, where only the first word of a sentence is capitalized.

Title caps exceptions are articles (a, an, the), coordinating conjunctions (and, but, for, not, or, so, yet), and prepositions with four or fewer letters (at, for, in, into). An example would be "*Neon Tetras in My Fish Tank.*"

Sentence caps exceptions are words that would be normally capitalized in text such as proper nouns or feature names. An example would be "*I want to visit Mt. Rainier in the springtime.*"

Maintain consistent capitalization practices to prevent a disjointed or jagged reading experience.

Use lowercase for:

- **Page titles**
- **List titles**
- **List group titles**
- **Push button control text or words that function as commands**
- **List items**
- **Example text that appears in search boxes**
- **Link controls in the middle of a sentence**

Use sentence caps for:

- **Check box and radio button labels**
- **Progress indicators**
- **Status, notification, and explanatory text**
- **Toggle switches**

Use all caps for:

- **Application titles**
- **Dates and times**
- **AM or PM**

Text guidelines – punctuation

… & . ? ! ( )

Reading UI text that has no punctuation or poor punctuation can lead to severe user confusion and frustration. Punctuation helps to clarify ambiguous sentences, places emphasis where it needs it, and provide hints to the reader about the context of the words they read.

Compare "*Coconuts healthy organic and delicious*" to "*Coconuts: healthy, organic, and delicious.*" and "*Slowly quietly unbuckle it cables kick*" to "*Slowly & quietly unbuckle it – cables kick!*" In these examples, punctuation reveals a tasty snack hidden in nonsense and emphasizes a potential danger.

Maintain consistent punctuation to prevent the user from becoming confused, clarify ambiguous text, and provide direct emphasis as needed.

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 Downloaded: XX%
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 an ellipsis in headings. • Do not use an ellipsis in button labels.
End punctuation (. ? !)	• Use end punctuation only in 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 using a parenthesis in the UI if possible, but use a parenthesis if you need to include an acronym or other short piece of information.

Miscellaneous

Windows Phone 7 has other hardware features available not previously mentioned, including:

- Bluetooth
- Camera flash
- Camera LED
- Micro SD
- Micro USB
- Wi-fi

All of these features have no direct UI components and developers will need to create custom UI elements if they need to represent them within their application.


Windows Phone