



Writing Great Windows Store Apps in XAML and C#

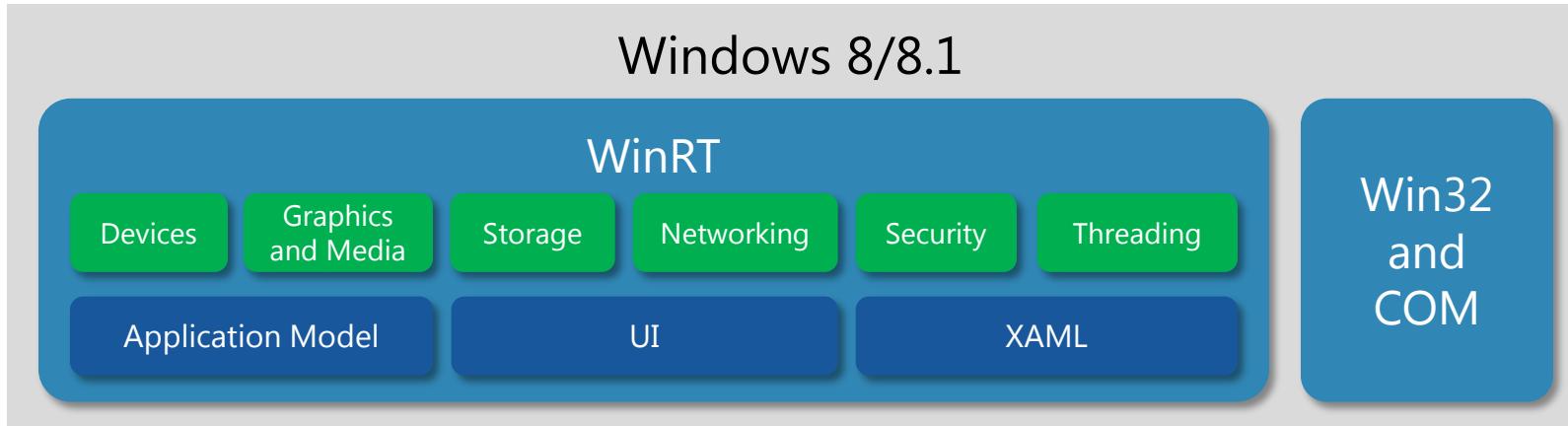
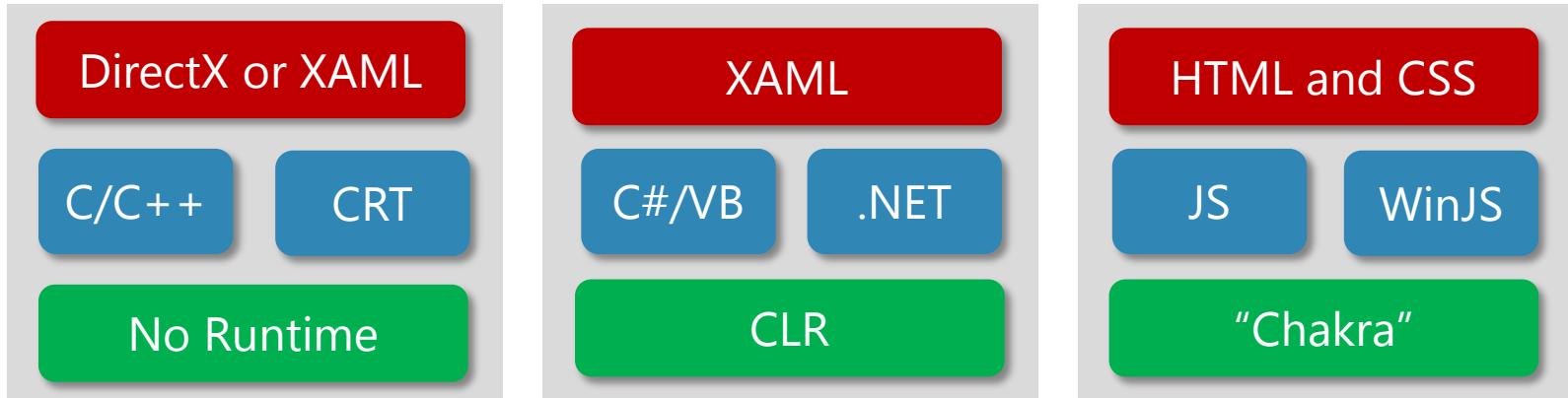
Jeff Prosise

jeffpro@wintellect.com

Windows Store App Model

- Focuses on reliable, secure, low-power apps running on PCs and portable devices
- Apps occupy all or part of the screen and are always single-instance
- Apps are suspended when not in the foreground and can be terminated at any time by the operating system
- Apps are distributed via Windows Store
 - Side-loading permitted for enterprises and developers
- Apps rely primarily on WinRT APIs

Windows Store App Architecture



The Windows Runtime (WinRT)

- Win32 API is old, complex, and built for C/C++
- WinRT is smaller, simpler, and multilingual
 - API is heavily asynchronous (don't block the UI thread!)
 - WINMDs contain metadata for language projections
 - Uses COM under the covers
- For languages, "use what you know"
 - Create WinRT components with C++ and C#
 - Consume WinRT components in all languages

<http://msdn.microsoft.com/en-us/library/windows/apps/br211377>

Consuming WinRT from C#

```
using Windows.Storage;
using Windows.UI.Popups;
.
.
.
// Read and display the contents of data.txt
var result = await PathIO.ReadTextAsync("ms-appx:///data.txt");
var dialog = new MessageDialog(result);
dialog.ShowAsync();
```

Consuming WinRT from JavaScript

```
var storage = Windows.Storage;
var popups = Windows.UI.Popups;
.
.
.
// Read and display the contents of data.txt
storage.PathIO.readTextAsync("ms-appx:///data.txt")
.then(function (result) {
    var dialog = new popups.MessageDialog(result);
    dialog.showAsync();
});
```

.NET for Windows Store Apps

- Supplements WinRT APIs for C# apps
 - Collections: Generic, ObjectModel, Concurrent
 - LINQ: Objects, XML, Parallel
 - Strings: StringBuilder, Encodings, Regex
 - IO: Binary/Stream/String/TextReader/Writer, Compression
 - Reflection and dynamic
 - Threading: Synchronization, Parallel, Tasks
 - Serialization and networking
- Subset of .NET 4.5 types and APIs

<http://msdn.microsoft.com/en-us/library/windows/apps/br230302.aspx>

WinJS for Windows Store Apps

- Windows Library for JavaScript
- Supplements WinRT APIs for JavaScript apps
 - HTTP networking
 - Classes and namespaces
 - Pages and navigation
 - Controls and data binding
 - Utility functions, promises, and more
- Contained in base.js and ui.js

<http://msdn.microsoft.com/en-us/library/windows/apps/br211377.aspx>

DEMO

WinRT in Action

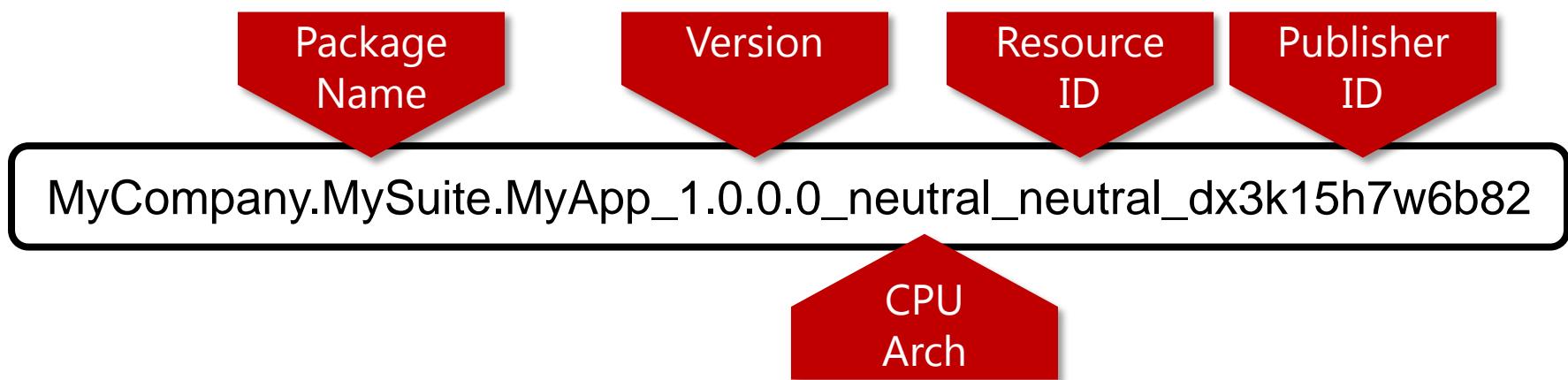
The App Package

- Build creates APPX file ("package") containing app's code, resources, manifest, and other files
 - APPX file is zip file signed with certificate in project's PFX
- Each package has two names
 - Full name (version- and CPU-specific)
 - Family name (version- and CPU-neutral)
- Windows 8.1 adds support for app bundles
 - Bundle can contain packages for different CPUs
 - Bundle can also contain resource packages

<http://msdn.microsoft.com/en-us/library/windows/apps/hh464929.aspx>

Package Full Name

- Generated from package name, version number, CPU architecture, resource ID, and publisher ID
 - Version and CPU architecture specified in Create App Packages wizard; resource ID currently unused



Package Family Name

- Generated from package name in manifest and publisher ID in certificate (Base32-encoded)

Application UI	Capabilities	Declarations	Packaging
Use this page to set the properties that identify and describe your package when it is deployed.			
Package Name:	MyCompany.MySuite.MyApp		
Package Display Name:	(Shown in Windows Store)		
Logo:	Assets\Windows Store Logo (50x50).png	X	Browse...
		Required Size : 50 x 50 pixels	
Publisher:	CN=Wintellect LLC		
Publisher Display Name:	Wintellect Corporation		
Package Family Name:	MyCompany.MySuite.MyApp_dx3k15h7w6b82		

Package
Name

Publisher
ID

Staging and Registration

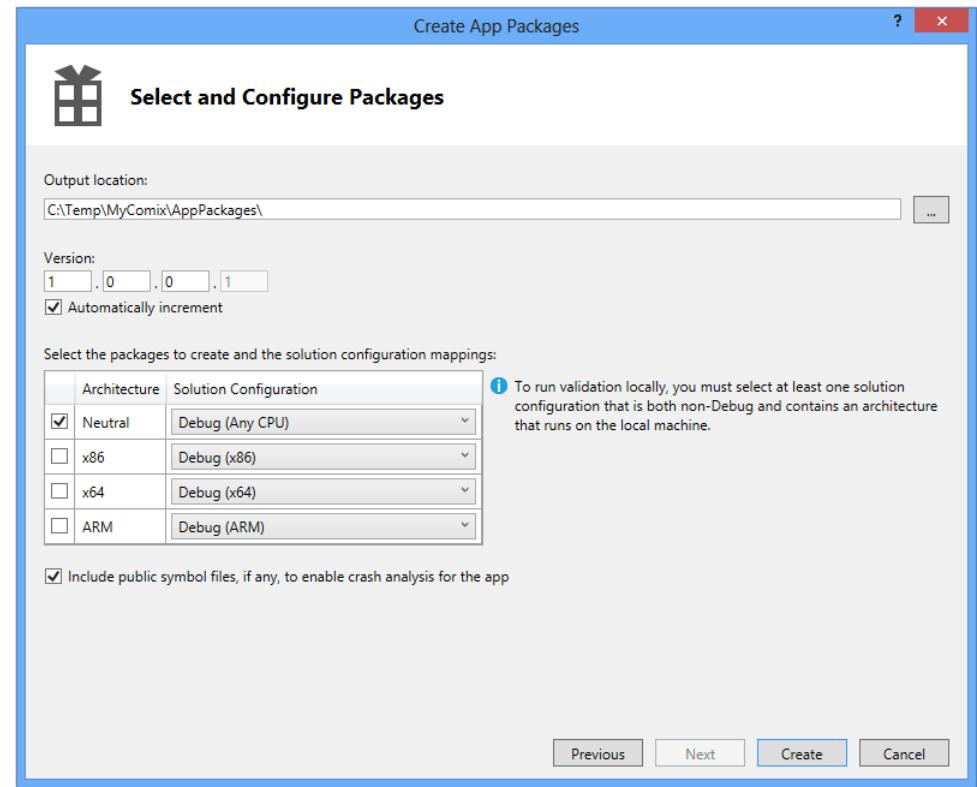
- Staging (once per machine)
 - Creates C:\Program Files\WindowsApps\{FullName} folder and copies unzipped APPX contents into it
 - Deleted when last user uninstalls the app
- Registration (once per user)
 - Creates %UserProfile%\AppData\Local\Packages\{FamilyName} folder to hold local application data
 - Also creates new keys in the registry for activation and contracts
 - Deleted when any user uninstalls the app
- Visual Studio registers but does not stage
 - Staging directory is project's build directory

Getting Package Information

```
// Get the package's installed location  
var path = Package.Current.InstalledLocation.Path;  
  
// Get the package's full name, family name, and name  
var full = Package.Current.Id.FullName;  
var family = Package.Current.Id.FamilyName;  
var name = Package.Current.Id.Name;  
  
// Get the package's publisher name and ID  
var publisher = Package.Current.Id.Publisher;  
var id = Package.Current.Id.PublisherId;  
  
// Get the package's major and minor version numbers  
var major = Package.Current.Id.Version.Major;  
var minor = Package.Current.Id.Version.Minor;
```

Creating an App Package

- Project -> Store -> Create App Packages
- Builds package files for distributing or uploading to the Windows Store
 - Includes PowerShell script for installing on other devices



<http://msdn.microsoft.com/en-us/library/windows/apps/hh975357.aspx>

Sideload (QA Deployments)

- Run Add-AppDevPackage.ps1 PowerShell script generated by wizard on destination device
 - Installs package (APPX)
 - Installs app's certificate in Trusted Peoples store
 - Acquires and installs free dev license if necessary
- Powershell script execution must be enabled
 - Set-ExecutionPolicy unrestricted

<http://msdn.microsoft.com/en-us/library/windows/apps/hh975356.aspx>

DEMO

Deploying to a Surface RT

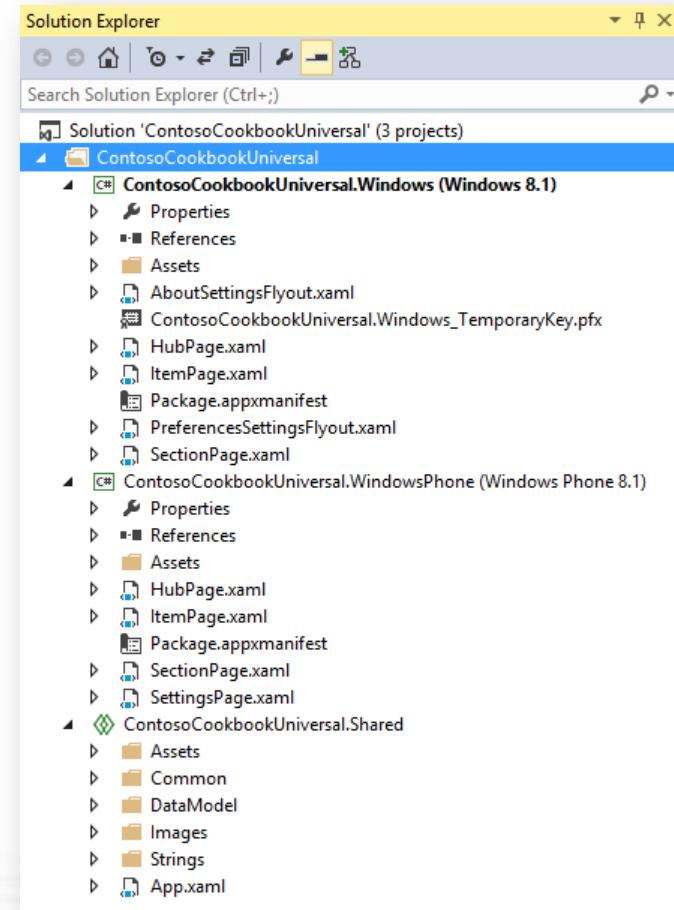
Universal Apps

- Announced at BUILD 2014
 - Require Visual Studio 2013 Update 2 (currently in RC)
- Run on Windows 8.1, Windows Phone 8.1, and...



Universal App Solution

- Windows 8.1 project
 - Contains "head" for Windows version of app
- Windows Phone 8.1 project
 - Contains "head" for Windows Phone version of app
- Shared project
 - Contains code, resources, and other assets shared by other projects
 - No target platform; files in shared project are shared with other projects via file linking



Shared Project

- Contains source code files, images, and other assets used in Windows and Windows Phone projects
 - Source code files must be able to compile in each project
- Cannot contain references to other assemblies
 - Add assembly references to Windows and Windows Phone projects
 - References can include platform-specific assemblies, Portable Class Libraries (PCLs), and even Windows Runtime components
- Most code can be shared between projects because Windows Phone 8.1 implements more than 90% of WinRT

WinRT vs. WinPRT

Windows (WinRT)

System.ServiceModel
System.ServiceModel.*
Windows.ApplicationModel.-
Contacts.Provider
Windows.ApplicationModel.-
Search
Windows.Data.Pdf
Windows.Devices.-
PointOfService
Windows.Devices.Scanners
Windows.Devices.Sms
Windows.Devices.Usb
Windows.Graphics.Printing
Windows.Media.Playlists
Windows.Security.Credentials.UI
Windows.Storage.BulkAccess
Windows.UI.ApplicationSettings
Windows.UI.Input.Inking
Windows.UI.Xaml.Printing

Windows Phone (WinPRT)

Windows.ApplicationModel.Chat
Windows.ApplicationModel.Email
Windows.ApplicationModel.Wallet
Windows.Media.Playback
Windows.Media.SpeechRecognition
Windows.ServicesMaps
Windows.UI.Xaml.ControlsMaps

A Venn diagram consisting of two overlapping circles. The left circle is light blue and labeled "Windows (WinRT)". The right circle is light red and labeled "Windows Phone (WinPRT)". The overlapping area is shaded in a darker maroon color and contains the text "Common APIs".

Common APIs

<http://firstfloorsoftware.com/Media/DiffLists/Windows%208.1-vs-Windows%20Phone%208.1.html>

DEMO

Universal Apps

Modern UIs

- XAML runtime contains numerous controls for building modern UIs
 - Pseudo-controls
 - Control-derivatives
 - ContentControl-derivatives
 - ItemsControl-derivatives
- Controls feature touch support, fluidity through built-in animations, layout awareness, and much more

Psuedo-Controls

Control	Description
Border	Draws borders around other elements
CaptureElement	Represents surfaces for showing video capture
Flyout	Represents flyout controls (8.1)
Image	Represents images
MediaElement	Represents surfaces for playing back video
MenuFlyout	Represents menu flyouts (8.1)
RichTextBlock	Represents blocks of rich text
RichTextBlockOverflow	Accepts text that overflows from RichTextBlocks
TextBlock	Represents blocks of text
ViewBox	Scales content to fit inside a defined area
WebView	Renders HTML and JavaScript content

MenuFlyout

```
<AppBarButton Icon="MailReply" Label="Reply">
  <AppBarButton.Flyout>
    <MenuFlyout>
      <MenuFlyoutItem Text="Reply" Click="OnReply" />
      <MenuFlyoutItem Text="Reply All" Click="OnReplyAll" />
      <MenuFlyoutItem Text="Forward" Click="OnForward" />
    </MenuFlyout>
  </AppBarButton.Flyout>
</AppBarButton>
```



Open File



Save



Reply

WebView

```
<WebView Source="http://msdn.microsoft.com/en-US/windows/apps" />
```



The screenshot shows the Microsoft Dev Center for Windows Store apps. The main header reads "Windows | Dev Center - Windows Store apps". The top navigation bar includes links for Home, Dashboard, Docs, Samples, Downloads, Support, and Community. A search bar is located in the top right corner. The main content area features a large image of a yellow submarine-like vehicle in water, with the text "Developing a Windows Store game? Start here" overlaid. Below this, a button says "Show me". To the left, a section titled "See how they did it" shows a hand interacting with a tablet displaying a food app interface. A caption below the tablet reads "Great apps are here. See the successes of Allrecipes and other app builders." A blue button at the bottom of this section says "Hear their stories". To the right, a "Popular" sidebar lists links such as "Windows Store apps: The basics", "Product guide for developers", "Downloads for developers", "Open a Windows Store developer account", "API reference", "Resources for iOS developers", "Develop games", "Sell your app", and "Forums". At the bottom, a "What's new" section lists "Building an end-to-end JavaScript app: Hilo", "Hilo sample app: C++", and "New Windows App Certification Kit".

DEMO

WebView Controls

Controls

Control	Description
DatePicker	Provides a UI for entering dates (8.1)
PasswordBox	Represents PasswordBox controls for password input
ProgressBar	Represents progress-bar controls for showing progress
ProgressRing	Represents progress-ring controls for indicating waits
RichEditBox	Represents rich text-editing controls
ScrollViewer	Represents ScrollViewer controls for scrolling and zooming
SemanticZoom	Represents SemanticZoom controls for semantic zoom
Slider	Represents slider controls for inputting values from ranges
TextBox	Represents TextBox controls for inputting text
TimePicker	Provides a UI for entering times (8.1)
ToggleSwitch	Represents toggle-switch controls for on-off operations

ToggleSwitch

```
<ToggleSwitch />      On
```



```
<ToggleSwitch OnContent="{x:Null}" OffContent="{x:Null}"  
Toggled="OnToggled" />
```



```
void OnToggled(object sender, RoutedEventArgs e)  
{  
    if (((ToggleSwitch)sender).IsOn)  
        // ToggleSwitch is on  
    else  
        // ToggleSwitch is off  
}
```

ProgressRing

```
<ProgressRing x:Name="Wait" Width="200" Height="200" />
```

```
// Show the ProgressRing
```

```
Wait.IsActive = true;
```

```
// Hide the ProgressRing
```

```
Wait.IsActive = false;
```



DEMO

Controls

Content Controls

Control	Description
AppBar	Represents application bars
Button	Represents buttons
CheckBox	Represents checkboxes
ComboBoxItem	Represents items in ComboBox controls
FlipViewItem	Represents items in FlipView controls
CommandBar	Represents command bars (8.1)
GridViewItem	Represents items in GridView controls
HyperlinkButton	Represents Hyperlink-buttons
ListBoxItem	Represents items in ListBox controls
ListViewItem	Represents items in ListView controls
RadioButton	Represents radio buttons
SettingsFlyout	Represents settings flyouts (8.1)

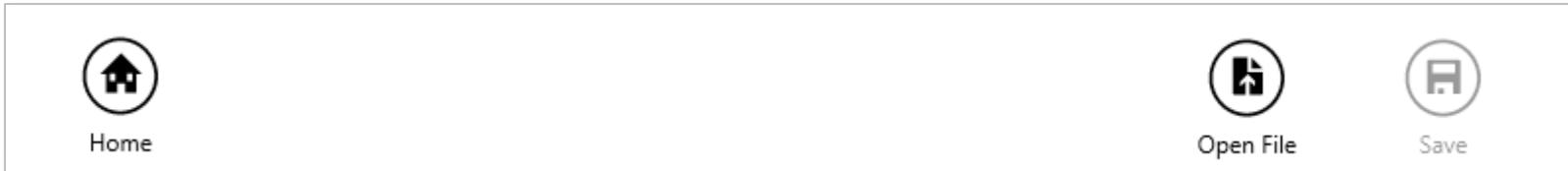
Application Bars/Command Bars

- Contain shortcuts to commonly used commands
 - Can appear at top and/or bottom of page
 - Divided into left and right commands
 - Displayed by swiping up or down, pressing Win+Z, or right-clicking
- Represented by AppBar class in Windows 8
- Superseded by CommandBar class in Windows 8.1
 - Contains AppBarButtons, AppBarToggleButtons, and AppBarSeparators
 - Automatically resizes if app resizes (IsCompact) and resizes buttons

<http://msdn.microsoft.com/en-us/library/windows/apps/hh465302>

CommandBar

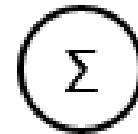
```
<Page.BottomAppBar>
  <CommandBar>
    <AppBarButton Icon="OpenFile" Label="Open" Click="OnOpen" />
    <AppBarButton Icon="Save" Label="Save" Click="OnSave"
      IsEnabled="False" />
    <CommandBar.SecondaryCommands>
      <AppBarButton Icon="Home" Label="Home" Click="OnHome" />
    </CommandBar.SecondaryCommands>
  </CommandBar>
</Page.BottomAppBar>
```



AppBarButton Icons

- AppBarButton.Icon property accepts SymbolIcons, FontIcons, BitmapIcons, and PathIcons

```
<AppBarToggleButton Label="FontIcon" Click="OnClick">
    <AppBarToggleButton.Icon>
        <FontIcon FontFamily="Candara" Glyph="&#x03A3;" />
    </AppBarToggleButton.Icon>
</AppBarToggleButton>
```



DEMO

Command Bars

Items Controls

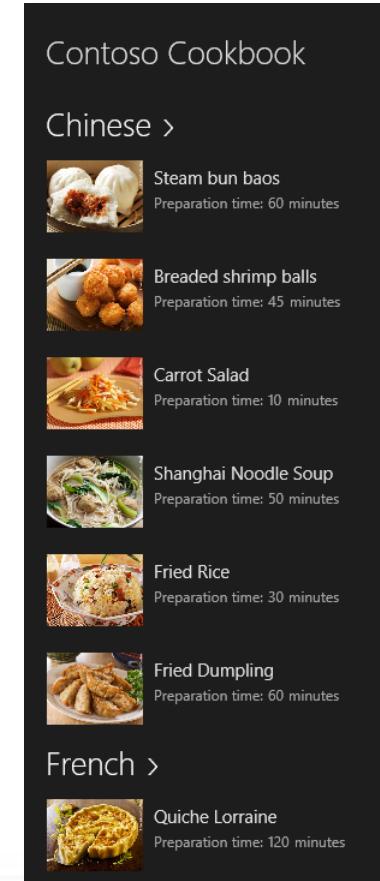
Control	Description
ComboBox	Represents ComboBox controls
FlipView	Represents FlipView controls
GridView	Represents GridView controls
Hub	Provides container for individually templated HubSections (8.1)
ListBox	Represents ListBox controls
ListView	Represents ListView controls

FlipView

- Displays horizontally scrolling list of items
 - FlipViewItem class models items
 - Items can be statically declared or created through data binding
- Fires SelectionChanged events
 - Fired when new item scrolls into view
 - Use SelectedIndex or SelectedItem property to identify item selected
- Derives from Selector (ItemsControl derivative)
- Enhanced in Windows 8.1 to support smooth scrolling with mouse

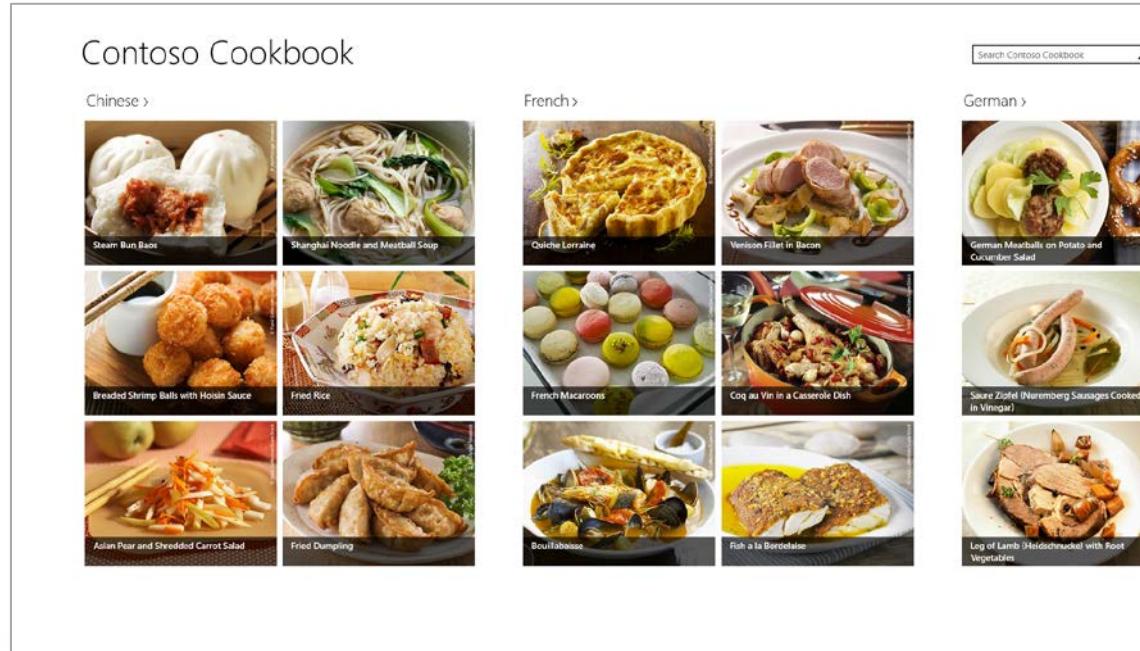
ListView

- Displays scrolling rows or columns of items
- Supports single and multiple selection
 - SelectionMode="None" turns selection off
- Optionally fires ItemClick events
 - Set IsItemClickEnabled="True"
- Supports data binding and data grouping
- Compatible with SemanticZoom control
- Great for displaying content when app is running in a narrow slice of the screen



GridView

- Presents horizontally scrolling groups of item grids, each from a common data source (if control is data-bound)



Hub

- Presents horizontally scrolling hub-sections, each templated and data-bound separately



Most Recent >



Mastering the Chrome Developer Tools
The Chrome Developer Tools are a developer's best friend when it comes to spying on network traffic, accessing local storage, auditing page...



Web Runtime Architecture
In the inaugural video in his series on Web performance, IIS Program Manager Tobin Titus takes you under the cover of your browser and...



Build Your First iOS App
Explore the exciting world of native iOS development using Xcode and Objective-C. This fast-paced session, the first in a series on iOS d...



Delicious iPhone Battery
Want fast?



Migrating ASP.NET Web Forms to MVC
Whether you're on the fence about migrating to ASP.NET MVC or you're already on the way, this course will help you determine if and how to pr...



Authoring jQuery Plugins
One of the most powerful aspects of jQuery is extensibility. The plugin architecture allows anyone to build components that extend jQuery...



CSS3 Responsive Design
In addition to being a popular buzzword in website design today, responsive design is focused on creating websites that respond to th...



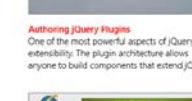
Microsoft Fakes
Microsoft Fakes is a tool that helps you to isolate code from its dependencies within your unit tests. Learn how to use it to test an ASP.NET MV...

Most Popular >



Fundamentals of AngularJS
AngularJS is a JavaScript framework that enables you to teach HTML new tricks. The goal of the framework is to enable a clean separation of c...





Authoring jQuery Plugins
One of the most powerful aspects of jQuery is extensibility. The plugin architecture allows anyone to build components that extend jQuery...



Reforming I/O-based Asynchronous Operations
Most applications are I/O bound. This video explains how to leverage asynchronous I/O in order to reduce resource overhead and how t...

DEMO

Items Controls

Orientation

- Many devices can sense 3D spatial orientation with high accuracy via onboard accelerometers
- Usually all we care about is "simple" orientation
 - Is device in portrait mode or landscape mode?
 - Is it face up or face down?
- Sometimes we want more
 - Labyrinth-type games that simulate physics
- By default, Windows 8 automatically rotates display when orientation-aware device rotates

Auto-Rotation Preferences

- You can restrict the orientations an app supports by specifying auto-rotation preferences
 - Declaratively through application manifest
 - Programmatically through `DisplayProperties.AutoRotationPreferences` property
- Display will only rotate when the orientation matches one of the preferences you specify
- Simulator doesn't honor these preferences
- Some devices only support landscape

Specifying Preferences Declaratively

Application Visual Assets Capabilities Declarations Content URLs Packaging

Use this page to set the properties that identify and describe your app.

Display name: ViewDemo

Entry point: ViewDemo.App

Default language: en-US [More information](#)

Description: ViewDemo

Supported rotations: An optional setting that indicates the app's orientation preferences.

 Landscape  Portrait  Landscape-flipped  Portrait-flipped



Support landscape and landscape-flipped only (do not rotate display when device rotates to portrait or portrait-flipped)

Specifying Preferences Programmatically

```
// Indicate that the display should not rotate when the  
// device rotates to portrait or portrait-flipped  
DisplayProperties.AutoRotationPreferences =  
    DisplayOrientations.Landscape |  
    DisplayOrientations.LandscapeFlipped;
```

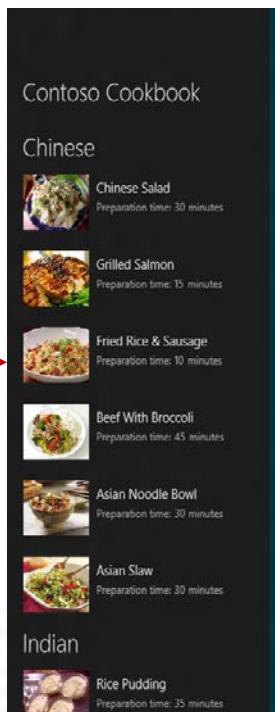
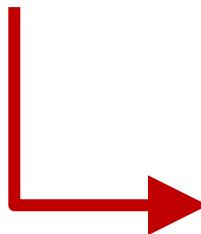
DEMO

Auto-Rotation Preferences

Snapping (Windows 8)

- Lets two apps run side by side
- Requires minimum screen resolution of 1,366 x 768

*Snapped
(320 pixels)*



Building Windows 8

An inside look from the Windows engineering team.

[Excerpt View](#) [Full Post View](#)

Running the Consumer Preview: system recommendations

Published 6 days ago by Steven Sinofsky

470 comments

We're very excited to get to the point in the project where we can provide a pre-release version of Windows 8 that is broadly usable by the tech community on a daily basis. We know folks who are anxious to run the Windows 8 Consumer Preview are probably interested in suggestions around what hardware to use. This post provides the technical details behind system recommendations. These are not system requirements, and are not final, but simply a view of what works best for running the Consumer Preview...

Welcome to Windows 8 – The Consumer Preview

Published 6 days ago by Steven Sinofsky

625 comments

Today is a big day for the Windows team. At Mobile World Congress in Barcelona, Spain a few moments ago, we unveiled the Windows 8 Consumer Preview to our partners and press. Based on a broad range of feedback, we have made over 100,000 code changes and the Consumer Preview represents a refined product ready for broad and daily usage by those of you willing to test a pre-release OS. You can download the Consumer Preview starting now at <http://preview.windows.com> . If you tried the Windows 8 Developer...

Using the language you want

Published 14 days ago by Steven Sinofsky

263 comments

Since its introduction in Windows 2000, Multilingual User Interface technology, or MUI, has allowed customers to install additional display languages on their Windows PCs and to switch between them. But for the majority of users, the language you got when you booted up your Windows PC for the first time was likely the one you were stuck with. For Windows 8, we have reimaged the display language experience, focusing on making additional display languages available to all Windows users, making them...

Languages

English
Français
Deutsch
Português (Brasil)
한국어
日本語
简体中文
Русский

Feeds & more

Blog Home
 Email Blog Author
 RSS for comments
 RSS for posts
 Atom

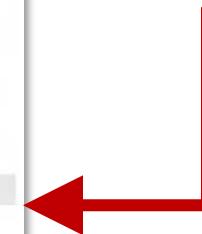
Search

 Search this blog Search all blogs

Blogs

Sign In

*Filled
(1024+ pixels)*



Resizing (Windows 8.1)

- Apps can be continuously resized to the desired width
 - Default minimum width is 500 pixels
 - Can be changed to 320 pixels through the manifest
- More than two apps can share the screen
- New ApplicationView properties reveal information about the view's current state
 - Orientation and IsFullScreen
 - AdjacentToLeftDisplayEdge and AdjacentToRightDisplayEdge
- Window size can be measured with Window.Bounds.Width and Window.Bounds.Height

Responding to Window-Size Changes

```
Window.Current.SizeChanged += (s, e) =>
{
    var orientation =
        ApplicationView.GetForCurrentView().Orientation;

    switch (orientation)
    {
        case ApplicationViewOrientation.Portrait:
            // Height is greater than width
            break;
        case ApplicationViewOrientation.Landscape:
            // Width is greater than height
            break;
    }
};
```

Determining the Window Size

```
Window.Current.SizeChanged += (s, e) =>
{
    var width = e.Size.Width;
    var height = e.Size.Height;

    if (width < 400.0)
    {
        // TODO: Adjust for narrow window
    }
    else
    {
        // TODO: Adjust for wider window
    }
};
```

Visual State Manager

- Manages visual states of UI elements
 - Knows what states an element supports
 - Knows how each state differs visually from default state
 - Provides API for transitioning between states
 - Knows how to unwind state transitions
- Defines states declaratively (in XAML)
 - Conducive to tooling support and code generation
- `VisualStateManager.GoToState` method triggers state transition

Defining Visual States

```
<Rectangle x:Name="Rect" Width="300" Height="200" Fill="Red" />
...
<VisualStateManager.VisualStateGroups>
  <VisualStateGroup>
    <VisualState x:Name="Normal" />
    <VisualState x:Name="Highlighted">
      <Storyboard>
        <ObjectAnimationUsingKeyFrames
          Storyboard.TargetName="Rect"
          Storyboard.TargetProperty="Fill">
          <DiscreteObjectKeyFrame KeyTime="0" Value="Yellow"/>
        </ObjectAnimationUsingKeyFrames>
      </Storyboard>
    </VisualState>
  </VisualStateGroup>
</VisualStateManager.VisualStateGroups>
```

Triggering State Transitions

```
// Transition the rectangle to the "Highlighted" state  
VisualStateManager.GoToState(this, "Highlighted", false);  
  
// Return the rectangle to the "Normal" state  
VisualStateManager.GoToState(this, "Normal", false);
```

Swapping Entire Views

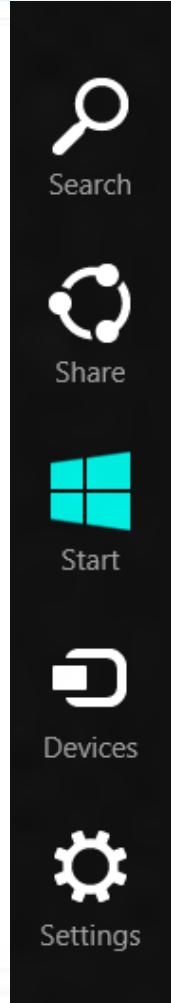
```
<VisualState x:Name="Narrow">
  <Storyboard>
    <ObjectAnimationUsingKeyFrames
      Storyboard.TargetName="Normal"
      Storyboard.TargetProperty="Visibility">
      <DiscreteObjectKeyFrame KeyTime="0" Value="Collapsed"/>
    </ObjectAnimationUsingKeyFrames>
    <ObjectAnimationUsingKeyFrames
      Storyboard.TargetName="Narrow"
      Storyboard.TargetProperty="Visibility">
      <DiscreteObjectKeyFrame KeyTime="0" Value="Visible"/>
    </ObjectAnimationUsingKeyFrames>
  </Storyboard>
</VisualState>
```

DEMO

Visual State Manager

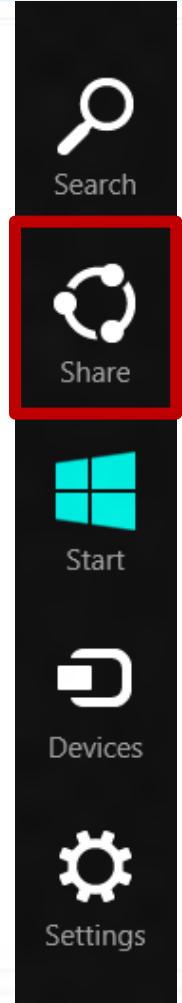
Charms

- Charms bar contains "charms"
 - Display by swiping from right or pressing Win-C
- Charms provide access to common features and promote a consistent, predictable UI
 - Search for content within your app (or other apps)
 - Share content from your app with other apps
 - Edit your app's configuration settings
 - Print, stream content to DLNA devices, and more
- Integrate with charms through contracts



Share Contracts

- Enable Windows Store apps to share content with other apps and consume shared content
- Share sources
 - Use DataTransferManager.DataRequested events to respond to share requests
 - Provide data using DataPackage methods
- Share targets
 - Registered via declaration in manifest
 - Activated via OnShareTargetActivated override
 - Consume data using DataPackageView methods



Handling DataRequested Events (8.1)

```
public ItemDetailPage()
{
    this.InitializeComponent();
    ...
    DataTransferManager.GetForCurrentView().DataRequested +=  
        OnDataRequested;
}  
  
void OnDataRequested(DataTransferManager sender,  
    DataRequestedEventArgs e)  
{  
    DataPackage dp = e.Request.Data;  
    // TODO: Use DataPackage methods to share data  
}
```

DataPackage Methods

Method	Description
SetBitmap	Makes a bitmap available for sharing
SetData	Makes data in an IRandomAccessStream available for sharing
SetDataProvider	Makes shared data available on demand through a data provider (delayed sharing)
SetHtmlFormat	Makes HTML data available for sharing
SetRtf	Makes rich text (RTF) data available for sharing
SetStorageItems	Makes files and folders available for sharing
SetText	Makes text available for sharing
SetUri	Shares content at the specified URI (deprecated in 8.1)
SetApplicationLink	Shares content in an app via URI (Windows 8.1)
SetWebLink	Shares content on the Web via URI (Windows 8.1)

Sharing Text

```
void OnDataRequested(DataTransferManager sender,  
    DataRequestedEventArgs e)  
{  
    var request = e.Request;  
    request.Data.Properties.Title = "Item title";  
    request.Data.Properties.Description = "Item description";  
    request.Data.SetText("Share this text with share targets");  
}
```

Sharing an Image

```
void OnDataRequested(DataTransferManager sender,
    DataRequestedEventArgs e)
{
    var request = e.Request;
    request.Data.Properties.Title = "Item title";
    request.Data.Properties.Description = "Item description";

    var uri = new Uri("ms-appx:///Images/Logo.png");
    var reference = RandomAccessStreamReference.CreateFromUri(uri);
    request.Data.Properties.Thumbnail = reference;
    request.Data.SetBitmap(reference);
}
```

Using Deferrals

```
async void OnDataRequested(DataTransferManager sender,
    DataRequestedEventArgs e)
{
    var request = e.Request;
    request.Data.Properties.Title = "Item title";
    request.Data.Properties.Description = "Item description";
    var deferral = request.GetDeferral();

    // TODO: Perform async work here, using C#'s await
    // keyword as needed to await async operations. Note
    // the async keyword decorating the method itself.

    deferral.Complete();
}
```

Displaying an Error Message

```
void OnDataRequested(DataTransferManager sender,  
    DataRequestedEventArgs e)  
{  
    var request = e.Request;  
    request.FailWithDisplayText("Unable to share");  
}
```

Displaying the Share UI

```
// Equivalent to the user selecting the Share charm  
DataTransferManager.ShowShareUI();
```

DEMO

Share Contracts

Search Contracts

- Enable Windows Store apps to expose search functionality through search pane
 - SearchPane class represents system's search pane
- Windows 8.1 deprecates this contract in favor of in-app search via SearchBox control
 - Provides search UI
 - Supports search suggestions and result suggestions
 - Fires search-related events
 - QueryChanged and QuerySubmitted
 - SuggestionsRequested and ResultSuggestionChosen



Supporting In-App Search

```
// XAML  
<SearchBox PlaceholderText="Search" Width="300" Height="32"  
QuerySubmitted="OnQuerySubmitted" />  
  
// C#  
void OnQuerySubmitted(SearchBox sender,  
    SearchBoxQuerySubmittedEventArgs args)  
{  
    var query = args.QueryText; // Search text  
    // TODO: Provide domain-specific search logic  
}
```



Providing Query Suggestions

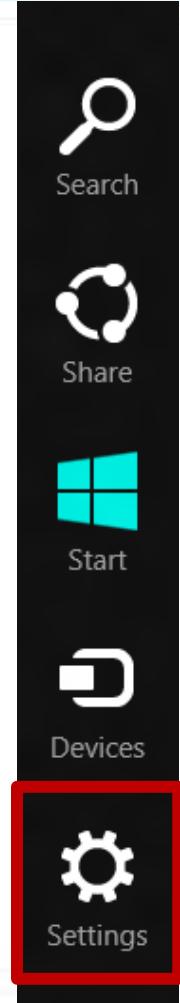
```
// XAML  
<SearchBox ... SuggestionsRequested="OnSuggestionsRequested" />  
  
// C#  
void OnSuggestionsRequested(SearchBox sender,  
    SearchBoxSuggestionsRequestedEventArgs args)  
{  
    var query = args.QueryText;  
    var suggestions = args.Request.SearchSuggestionCollection;  
  
    // _suggestions is array of strings containing possible  
    // query suggestions  
    var matches = _suggestions.Where(x => x.IndexOf(query,  
        StringComparison.OrdinalIgnoreCase) == 0);  
  
    suggestions.AppendQuerySuggestion(matches);  
}
```

DEMO

Search Contracts

Settings Contracts

- Enable Windows Store apps to expose configuration settings through settings pane
 - For example, language preference or toggle features on and off
- Integrate with Settings charm to implement consistent and predictable configuration UI
 - Handle `SettingsPane.CommandsRequested` events to add commands to Settings menu
 - Respond by adding `SettingsCommands` containing command text and command handlers
- Use `SettingsFlyout` control to create flyouts (8.1)



Adding Settings Commands

```
SettingsPane.GetForCurrentView().CommandsRequested += (s, e) =>
{
    // Add an About command
    var about = new SettingsCommand("about", "About", OnAbout);
    e.Request.ApplicationCommands.Add(about);

    // Add a Preferences command
    var pref = new SettingsCommand("preferences", "Preferences",
        OnPreferences);
    e.Request.ApplicationCommands.Add(pref);
};
```

Responding to Settings Commands

```
// Handler for the About command  
void OnAbout (IUICommand command)  
{  
    new AboutSettingsFlyout().Show();  
}
```

```
// Handler for the Preferences command  
void OnPreferences (IUICommand command)  
{  
    new PreferencesSettingsFlyout().Show();  
}
```

Application Data Stores

- Windows.Storage.ApplicationData class provides properties referencing app data stores for persistent storage of settings

Property	Description
LocalFolder	Returns StorageFolder reference to root folder in local data store
LocalSettings	Returns reference to settings container in local data store
RoamingFolder	Returns StorageFolder reference to root folder in roaming data store
RoamingSettings	Returns reference to settings container in roaming app data store
TemporaryFolder	Returns StorageFolder reference to root folder in temporary data store

Using LocalSettings

```
// Write the preferred language to LocalSettings  
var settings = ApplicationData.Current.LocalSettings;  
settings.Values["Language"] = "fr-fr";  
  
// Retrieve the preferred language from LocalSettings  
var settings = ApplicationData.Current.LocalSettings;  
if (settings.Values.ContainsKey("Language"))  
    string language = (string)settings.Values["Language"];  
  
// Remove the preferred language from LocalSettings  
var settings = ApplicationData.Current.LocalSettings;  
settings.Values.Remove("Language");
```

Using RoamingSettings

```
// Write the preferred language to RoamingSettings  
var settings = ApplicationData.Current.RoamingSettings;  
settings.Values["Language"] = "fr-fr";  
  
// Retrieve the preferred language from RoamingSettings  
var settings = ApplicationData.Current.RoamingSettings;  
if (settings.Values.ContainsKey("Language"))  
    string language = (string)settings.Values["Language"];  
  
// Remove the preferred language from RoamingSettings  
var settings = ApplicationData.Current.RoamingSettings;  
settings.Values.Remove("Language");
```

DEMO

Settings Contracts

Tiles

- Windows 8.1 supports four tile sizes
 - Small (70x70), medium (150x150), wide (310x150), and large (310x310)
 - User can switch between sizes on start screen
 - 150x150 tile is required; all others optional
- Apps can create secondary tiles programmatically
- Tiles can show live content and badges

Medium Tile



Wide Tile



Live Tile with Badge



Badge Updates

- Show numbers or glyphs on a tile to alert the user to new content, changed state, etc.
 - Glyph: none, activity, alert, available, away, busy, newMessage, paused, playing, unavailable, error, attention
 - Number: 1 – 99
- Perform immediate updates or timed updates
- Relevant Windows.UI.Notifications classes
 - BadgeUpdateManager
 - BadgeUpdater

Displaying a Numeric Badge

```
var xml = BadgeUpdateManager.GetTemplateContent  
    (BadgeTemplateType.BadgeNumber); // <badge value="" />  
((XmlElement)xml.GetElementsByTagName("badge")[0]).  
    SetAttribute("value", "7");  
var bu = BadgeUpdateManager.CreateBadgeUpdaterForApplication();  
bu.Update(new BadgeNotification(xml));
```



Displaying a Glyph Badge

```
var xml = BadgeUpdateManager.GetTemplateContent  
    (BadgeTemplateType.BadgeGlyph); // <badge value="" />  
((XmlElement)xml.GetElementsByTagName("badge")[0]).  
    SetAttribute("value", "newMessage");  
var bu = BadgeUpdateManager.CreateBadgeUpdaterForApplication();  
bu.Update(new BadgeNotification(xml));
```



Clearing a Badge

```
var bu = BadgeUpdateManager.CreateBadgeUpdaterForApplication();  
bu.Clear();
```

DEMO

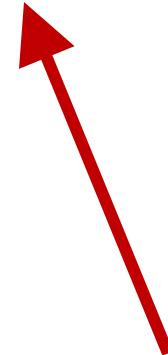
Badge Updates

Tile Updates

- Update text and images on a tile to make the tile seem alive and engaging
 - Updates can be immediate, periodic, or scheduled
 - Updates can also be queued, allowing system to cycle between tile updates automatically
- Relevant Windows.UI.Notifications classes
 - TileUpdateManager and TileUpdater
 - TileNotification and ScheduledTileNotification
- Updates can be disabled by user

Determining Whether Updates are Enabled

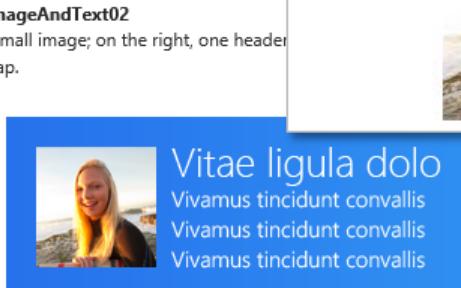
```
var tu = TileUpdateManager.CreateTileUpdaterForApplication();  
  
if (tu.Settings == NotificationSetting.Enabled)  
{  
    // Tile updates are enabled  
}
```



*Enabled
DisabledByUser
DisabledByGroupPolicy
DisabledByManifest*

Tile Templates

- `TileTemplateType` enumeration contains 72 values (46 in Windows 8), each representing a different XML tile format

<p>TileSquareText04 One string of regular text wrapped over a maximum width.</p> <p>Example</p>  <p>Vivamus tincidunt convallis urna, in ultrices nisi euismod sed. Aenean vitae ligula id dolor aliquam adipiscing. Praesent cursus diam sapien, hendrerit varius lectus.</p>	<p>TileWideImageCollection One large square image with four smaller square images to its right, no text.</p> <p>Example</p>  <p>Vivamus tincidunt convallis urna, in ultrices nisi euismod sed. Aenean vitae ligula id dolor aliquam adipiscing. Praesent cursus diam sapien, hendrerit varius lectus.</p>	<p>TileWideSmallImageAndText02 On the left, one small image; on the right, one header and two lines of text.</p> <p>Text does not wrap.</p> <p>Example</p>  <p>Vitae ligula dolo Vivamus tincidunt convallis Vivamus tincidunt convallis Vivamus tincidunt convallis</p>	<p>TileWidePeekImageAndText01 Top: One wide image. Bottom: One string of regular text wrapped over a maximum width.</p> <p>Example</p>  <p>Vivamus tincidunt convallis urna, in ultrices nisi euismod sed. Aenean vitae ligula id dolor aliquam adipiscing. Praesent cursus diam sapien, hendrerit varius lectus.</p>
--	---	---	--

<http://msdn.microsoft.com/en-us/library/windows/apps/hh761491.aspx>

Updating a Tile

```
var xml = TileUpdateManager.GetTemplateContent  
    (TileTemplateType.TileWideImageAndText01);  
((XmlElement)xml.GetElementsByTagName("image")[0]).  
    SetAttribute("src", "ms-appx:///Assets/Caprese.jpg");  
xml.GetElementsByTagName("text")[0].InnerText = "Caprese salad";  
var tu = TileUpdateManager.CreateTileUpdaterForApplication();  
tu.Update(new TileNotification(xml));
```



Clearing a Tile

```
var tu = TileUpdateManager.CreateTileUpdaterForApplication();  
tu.Clear();
```

Queued Tile Updates

- A tile can cycle through last five updates
 - Disabled by default
 - Enabled with `TileUpdater.EnableNotificationQueue`
- By default, when queue is full, a new update discards the oldest
 - However, you can assign a tag to each update to replace a specific update with a new one
- Useful for news feeds, stocks, weather, etc.

Enabling Queued Tile Updates

```
var tu = TileUpdateManager.CreateTileUpdaterForApplication();  
tu.EnableNotificationQueue(true);
```

Queuing Tile Updates

```
var xml = TileUpdateManager.GetTemplateContent  
    (TileTemplateType.TileWideImageAndText01);  
  
// Queue the first update  
((XmlElement)xml.GetElementsByTagName("image")[0]).  
    SetAttribute("src", "ms-appx:///Assets/Caprese.jpg");  
xml.GetElementsByTagName("text")[0].InnerText = "Caprese salad";  
tu.Update(new TileNotification(xml));  
  
// Queue the second update  
((XmlElement)xml.GetElementsByTagName("image")[0]).  
    SetAttribute("src", "ms-appx:///Assets/Spaghetti.jpg");  
xml.GetElementsByTagName("text")[0].InnerText = "Spaghetti";  
tu.Update(new TileNotification(xml));
```

DEMO

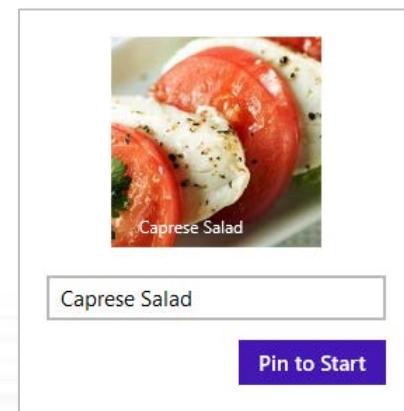
Tile Updates

Secondary Tiles

- Allow users to launch apps into a "bookmarked" state
 - e.g., a specific stock, location, or contact
 - Must be created programmatically
- Windows.UI.StartScreen.SecondaryTile class represents secondary tiles
 - Arguments property passed to app on activation
 - Other properties specify tile's ID and visual attributes
 - RequestCreateAsync method creates a secondary tile
- User must approve tile creation and can unpin a secondary tile at any time

Creating a Secondary Tile (Windows 8.1)

```
var tile = new SecondaryTile(  
    "1000",                                     // Tile ID  
    "Caprese Salad",                            // Tile display name  
    "Caprese",                                  // Activation argument  
    new Uri("ms-appx:///Assets/Square.jpg"), // Tile image URI  
    TileSize.Square150x150                      // Tile size  
);  
  
tile.VisualElements.ShowNameOnSquare150x150Logo = true;  
await tile.RequestCreateAsync();
```



Consuming Tile Arguments

```
// In App.xaml.cs
protected override void OnLaunched(LaunchActivatedEventArgs args)
{
    .
    .
    .
    // If the app was activated from a tile, navigate to the start
    // page and pass in the tile arguments
    if (!String.IsNullOrEmpty(args.Arguments))
    {
        rootFrame.Navigate(typeof(MainPage), args.Arguments);
    }
}
```

Updating a Secondary Tile

```
var xml = TileUpdateManager.GetTemplateContent  
    (TileTemplateType.TileWideImageAndText01);  
((XmlElement)xml.GetElementsByTagName("image")[0]).  
    SetAttribute("src", "ms-appx:///Assets/Caprese.jpg");  
xml.GetElementsByTagName("text")[0].InnerText = "Caprese salad";  
  
var tu = TileUpdateManager.CreateTileUpdaterForSecondaryTile(id);  
tu.Update(new TileNotification(xml));
```



Tile ID

Badging a Secondary Tile

```
var xml = BadgeUpdateManager.GetTemplateContent  
    (BadgeTemplateType.BadgeGlyph);  
((XmlElement)xml.GetElementsByTagName("badge")[0]).  
    SetAttribute("value", "newMessage");  
var bn = new BadgeNotification(xml);  
  
var bu = BadgeUpdateManager.CreateBadgeUpdaterForSecondaryTile(id);  
bu.Update(bn);
```



Tile ID

DEMO

Secondary Tiles

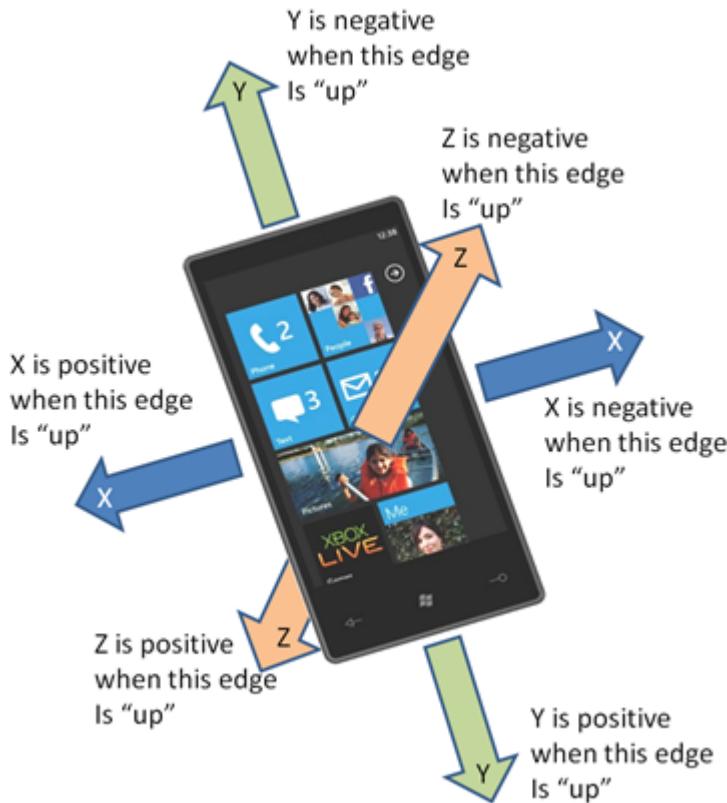
Sensor APIs

- Windows.Devices.Sensors namespace includes types providing assorted sensor APIs
 - Accelerometer
 - Compass
 - Gyrometer
 - Inclinometer
 - LightSensor
 - OrientationSensor and SimpleOrientationSensor
- Sensor APIs require working sensors (obviously!)

Accelerometer

- Most mobile devices have accelerometers
- Accelerometer class provides acceleration data in three dimensions (X, Y, and Z) for sensing device orientation in real time
 - GetCurrentReading for one-time reading
 - ReadingChanged events for stream of readings
 - Fired only when an acceleration vector changes!
- Accelerometer.Shaken event fires when device is shaken

Acceleration Vectors



Getting Oriented

- $X=0, Y=0$, and $Z=-1$ means device is face up
- $X=0, Y=-1, Z=0$ means device is standing up
- $X=-1, Y=0, Z=0$ means device is on left side
- $X=0, Y=0, Z=0$ means...

Detecting an Accelerometer

```
if (Accelerometer.GetDefault() != null)
{
    // Accelerometer present
}
```

Getting an Accelerometer Reading

```
var accelerometer = Accelerometer.GetDefault();  
  
if (accelerometer != null)  
{  
    var accel = accelerometer.GetCurrentReading();  
    var ax = accel.AccelerationX;  
    var ay = accel.AccelerationY;  
    var az = accel.AccelerationZ;  
}
```

Detecting Accelerometer Changes

```
if (accelerometer != null)
{
    accelerometer.RadingChanged += (s, e) =>
    {
        // WARNING: Not on the UI thread!
        var ax = e.Rading.AccelerationX;
        var ay = e.Rading.AccelerationY;
        var az = e.Rading.AccelerationZ;
    };
}
```

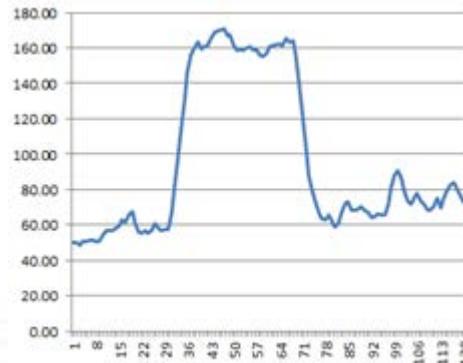
Using the Minimum Report Interval

```
// Get the device's minimum report interval  
var min = accelerometer.MinimumReportInterval;  
  
// Request a maximum of 20 readings per second  
accelerometer.ReportInterval = 50; // 50 millisecond intervals  
  
// Request default interval for device  
accelerometer.ReportInterval = 0;
```

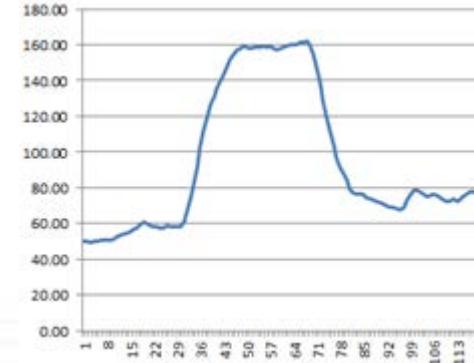
Smoothing Accelerometer Data

- Accelerometer data stream is noisy ("jittery")
- Apply smoothing algorithm to eliminate jitters
 - e.g., data averaging or low-pass filtering
- http://windowsteamblog.com/windows_phone/b/wpdev/archive/2010/09/08/using-the-accelerometer-on-windows-phone-7.aspx

Raw Data



Smoothed Data



Detecting Shakes

```
accelerometer.Shaken += (s, e) =>
{
    // Device was shaken
};
```

DEMO

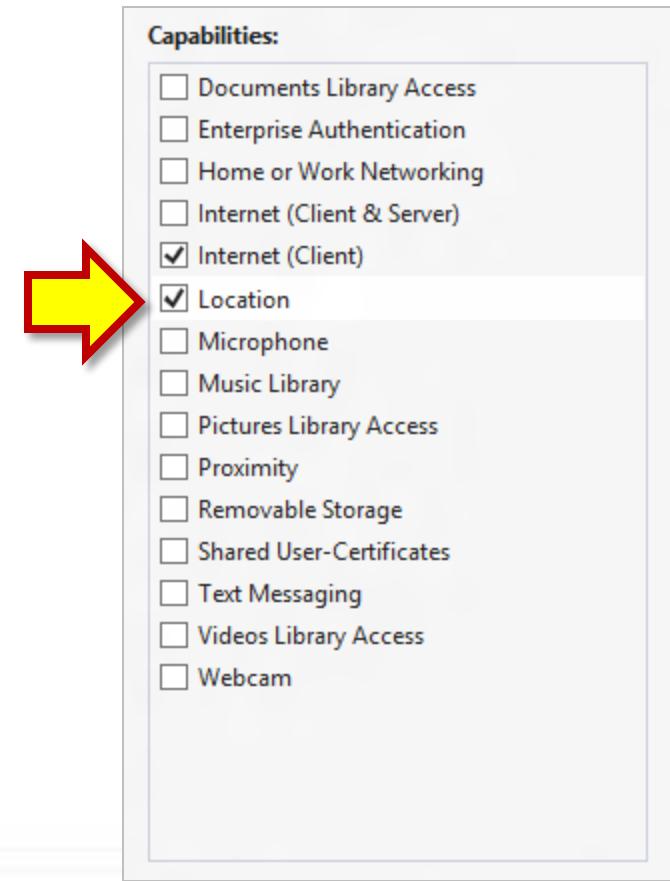
Accelerometer Input

Location API

- Windows.Devices.Geolocation.Geolocator class provides API for building location-aware apps
 - GetGeopositionAsync retrieves current location
 - PositionChanged event allows tracking of location
- Location API abstracts source of location data
 - GPS (3 to 7 meters)
 - WiFi positioning (10 meters to several miles)
 - Cell-tower triangulation (200 to 1,000 meters)
- Does not require a mobile device!

Location Capability

- Apps that use the location API must declare as much in their manifests
- WinRT throws exception if location API is called without this capability



Getting the Current Location

```
try
{
    var locator = new Geolocator();
    var location = await locator.GetGeopositionAsync();
    var latitude = location.Coordinate.Latitude;
    var longitude = location.Coordinate.Longitude;
    var altitude = location.Coordinate.Altitude;
    var accuracy1 = location.Coordinate.Accuracy;
    var accuracy2 = location.Coordinate.AltitudeAccuracy;
    var source = location.Coordinate.PositionSource;
}
catch (System.UnauthorizedAccessException ex)
{
    // Capability not declared or user declined permissions prompt
}
catch (TaskCanceledException)
{
    // Operation timed out
}
```

Requesting High Accuracy

```
var locator = new Geolocator();
locator.DesiredAccuracy = PositionAccuracy.High;
var location = await locator.GetGeopositionAsync();
```

Specifying Position Options

```
var locator = new Geolocator();
var maxage = TimeSpan.FromSeconds(5);
var timeout = TimeSpan.FromSeconds(10);
var location = await locator.GetGeopositionAsync(maxage, timeout);
```

Tracking the Current Location

```
var locator = new Geolocator();

locator.PositionChanged += (s, e) =>
{
    var latitude = e.Position.Coordinate.Latitude;
    var longitude = e.Position.Coordinate.Longitude;
    var altitude = e.Position.Coordinate.Altitude;
    var heading = e.Position.Coordinate.Heading;
    var speed = e.Position.Coordinate.Speed;
};
```

Setting the Movement Threshold

```
var locator = new Geolocator();
locator.MovementThreshold = 10.0; // 10 meters

locator.PositionChanged += (s, e) =>
{
    var latitude = e.Position.Coordinate.Latitude;
    var longitude = e.Position.Coordinate.Longitude;
    var altitude = e.Position.Coordinate.Altitude;
    var heading = e.Position.Coordinate.Heading;
    var speed = e.Position.Coordinate.Speed;
};
```

Geofencing (Windows 8.1)

- API that notifies your app when the user moves into or out of a specified geographic area
- Windows.Devices.Geolocation.Geofencing classes expose API
 - Geofence class represents a geofence
 - Defined by latitude, longitude, and radius
 - Specifies when to fire notifications (entry, exit, and removal)
 - Supports optional start time, dwell time, and duration
 - GeofenceMonitor class contains collection of Geofences, fires GeofenceStateChanged events, and includes methods for obtaining reports when events fire
- Currently supports circular geofences only

Using Geofencing

```
Geocircle geocircle = new Geocircle(new BasicGeoposition
    { Latitude = 35.880511, Longitude = -84.200997 }, 10.0);
MonitoredGeofenceStates mask = MonitoredGeofenceStates.Entered |
    MonitoredGeofenceStates.Exited;
var fence = new Geofence("Starbucks", geocircle, mask, true);

GeofenceMonitor.Current.GeofenceStateChanged += (s, e) =>
{
    var reports = s.ReadReports();
    foreach (var report in reports)
    {
        if (report.Geofence.Id == "Starbucks" &&
            report.NewState == GeofenceState.Entered)
        {
            // Entered Starbucks!
        }
    }
};

GeofenceMonitor.Current.Geofences.Add(fence);
```

DEMO

Location API

Networking

- WinRT contains rich networking support
 - Network information and status - NetworkInformation class
 - HTTP networking - HttpClient
 - Sockets - DatagramSocket, StreamSocket, and StreamSocketListener
 - WebSockets - StreamWebSocket and MessageWebSocket
 - Easy OAuth authentication - WebAuthenticationBroker
 - Proximity networking (NFC) - ProximityDevice and PeerFinder
 - Bluetooth networking - BluetoothDevice
- Also contains support for generating and consuming data in popular formats (e.g., JSON)

HttpClient

- Windows.Web.Http class for HTTP networking
 - New in Windows 8.1
 - Supercedes Windows 8's System.Net.Http.HttpClient class
- Fetch content from the Web, call REST services, and more
 - Supports GET, POST, PUT, and DELETE
 - Supports chainable request filters
 - Supports progress events
 - Supports HTTPS
- No cross-domain/cross-origin restrictions

<http://msdn.microsoft.com/en-us/library/windows/apps/windows.web.http.httpclient.aspx>

Downloading an Image

```
// Fetch the image
var client = new HttpClient();
var response = await client.GetAsync(new Uri("..."));
var buffer = await response.Content.ReadAsBufferAsync();

// Convert the IBuffer into an IRandomAccessStream
var ras = new InMemoryRandomAccessStream();
await ras.WriteAsync(buffer);
ras.Seek(0);

// Show the image
var bitmap = new BitmapImage();
await bitmap.SetSourceAsync(ras);
Output.Source = bitmap; // "Output" is name of XAML Image
```

Calling a REST Service

```
var client = new HttpClient();
var response = await client.GetAsync(new Uri("..."));

if (response.IsSuccessStatusCode)
{
    // Assumes response contains a string (e.g., JSON)
    var result = await response.Content.ReadAsStringAsync();
}
```

Specifying a Time-Out

```
var cts = new CancellationTokenSource();
cts.CancelAfter(5000); // Wait up to 5 seconds

try
{
    var client = new HttpClient();
    var response =
        await client.GetAsync(new Uri("...")).AsTask(cts.Token);
}
catch(OperationCanceledException)
{
    // Operation timed out
}
```

WebAuthenticationBroker

- Windows.Security.Authentication.Web class that enables seamless authentication with online identity providers
 - AuthenticateAsync method authenticates with remote provider
 - Displays provider's login UI unless told not to
- Supports OAuth and OpenID protocols
 - Amazon, Dropbox, Evernote, Facebook, Flickr, FourSquare, GitHub, Google, Instagram, LinkedIn, Reddit, Tumblr, Twitter, Yammer, and many others
- Allows app to gain access to secure sites without having to solicit, transmit, or cache login credentials
 - Authentication token can be saved for future access

Authenticating with Facebook via OAuth

```
string auth = https://www.facebook.com/dialog/oauth?" +  
    "client_id={0}&redirect_uri={1}&scope={2}&response_type=token";  
  
string redirect =  
    "https://www.facebook.com/connect/login_success.html";  
  
string id = "..."; // Facebook app ID  
  
WebAuthenticationResult result =  
    await WebAuthenticationBroker.AuthenticateAsync(  
        WebAuthenticationOptions.None,  
        new Uri(String.Format(auth, id, redirect, "email"),  
            UriKind.Absolute),  
        new Uri(_redirect, UriKind.Absolute)  
    );
```

Retrieving a Facebook OAuth Token

```
if (result.ResponseStatus == WebAuthenticationStatus.Success)
{
    var data = result.ResponseData.Substring
        (result.ResponseData.IndexOf('#'));
    var values = data.Split('&');
    var token = values[0].Split('=')[1];    // Access token
    var seconds = values[1].Split('=')[1]; // Seconds valid
    var expiration = DateTime.UtcNow.AddSeconds(int.Parse(seconds));
    break;
}

// Other possible values for ResponseStatus include
// WebAuthenticationStatus.ErrorHttp and
// WebAuthenticationStatus.UserCancel
```

Using a Facebook OAuth Token

```
// Use Facebook's Graph API to retrieve user's e-mail address

var client = new HttpClient();
client.DefaultRequestHeaders.Authorization =
    new HttpCredentialsHeaderValue("OAuth", token);

var result = await client.GetStringAsync(new Uri(
    "https://graph.facebook.com/me?scope=email",
    UriKind.Absolute)
);

var profile = JsonObject.Parse(result);
var email = profile["email"].GetString();
```

DEMO

WebAuthenticationBroker

Windows Store App Lifecycle

- Windows Store apps run one at a time
 - Exception: Snapping allows two apps to run
- Apps not in the foreground are suspended
- Suspended apps can be terminated at any time
 - e.g., system needs the memory the app consumes
- If reactivated, a terminated app should appear as if it was not terminated
- Save state before app is suspended and restore it when app is reactivated

Application.Suspending

- Event fired by Windows.UI.Xaml.Application class before an app is suspended
 - Indicates app is about to be suspended
 - Also fires if app is closed by user
 - Allows you 5 seconds to complete your work
 - Wall-clock time in Windows 8, "resource time" in 8.1
- Handle Suspending events and save state in case app is terminated while suspended
 - Even the system doesn't know at suspension time whether the app will later be terminated
 - Assume it will be terminated!

Application.OnLaunched

- Virtual method called each time app is launched
- LaunchActivatedEventArgs.PreviousExecutionState reveals whether app was terminated
 - ApplicationExecutionState.Terminated indicates app was restarted following suspension and termination
 - Other possible values include Running, NotRunning, Suspended, and ClosedByUser
- If app was terminated, restore state saved when app was suspended

DEMO

Preserving State

Background Tasks

- Allow apps to execute code when not active
 - App is suspended, but background tasks execute
 - No UI permitted except for tiles, toasts, and badges
- Background tasks implement `IBackgroundTask`
 - Execute in response to triggers (e.g., `SystemTrigger`)
 - Tasks can optionally have conditions attached
- `BackgroundTaskBuilder` class provides API for registering background tasks

[http://msdn.microsoft.com/en-us/library/windows/apps/xaml/Hh977056\(v=win.10\).aspx](http://msdn.microsoft.com/en-us/library/windows/apps/xaml/Hh977056(v=win.10).aspx)

Background Task Constraints

- CPU utilization is limited
 - Applies to both AC and DC power
 - Lock-screen apps get 2 seconds every 15 minutes
 - Others receive 1 second every 2 hours
- Network utilization is limited
 - Limited on DC power; no limit for AC
 - Bandwidth varies depending on network hardware
 - For example, WiFi gets more bandwidth than mobile
- If quota is reached, task is temporarily suspended

Implementing a Background Task

- Add "Windows Runtime Component" project to solution and reference it from main project
- Add class that implements IBackgroundTask
 - Must be public and sealed
 - Override Run; use deferral if performing async calls
- Register background task from main app
 - First make sure it isn't already registered
 - Specify trigger and (optionally) conditions
- Add background-task declaration to manifest

Implementing IBackgroundTask

```
namespace SampleBackgroundTask
{
    public sealed class TimeZoneTask : IBackgroundTask
    {
        public void Run(IBackgroundTaskInstance taskInstance)
        {
            // TODO: Add code that executes in the background
        }
    }
}
```

Enumerating Registered Tasks

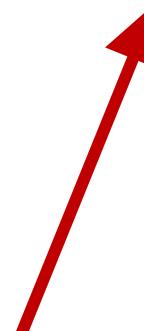
```
foreach (var task in BackgroundTaskRegistration.AllTasks)  
{  
    var name = task.Value.Name;  
}
```

Registering a Background Task

```
var builder = new BackgroundTaskBuilder();
builder.Name = "TimeZoneTask";
builder.TaskEntryPoint = "SampleBackgroundTask.TimeZoneTask";

builder.SetTrigger(
    new SystemTrigger(SystemTriggerType.TimeZoneChange, false)
);

builder.Register();
```



Recurring rather than one-shot

Declaring a Background Task

Application UI Capabilities Declarations **Packaging**

Use this page to add declarations and specify their properties.

Available Declarations: Background Tasks **Add**

Supported Declarations: Background Tasks **Remove**

Description:
Enables the app to specify the class name of an in-proc server DLL that runs the app code in the background in response to external trigger events. The class hosted in the in-proc server DLL is activated for background activation, and its Run method is invoked.
Multiple instances of this declaration are allowed in each app.
[More information](#)

Properties:

Supported task types

Audio
 Control channel
 System event
 Timer
 Push notification

App settings

Executable:

Entry point: `SampleBackgroundTask.TimeZoneTask`

Start page:

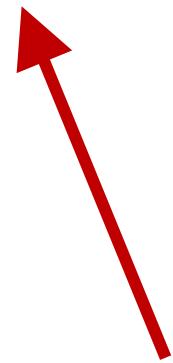
For System and Maintenance Triggers

Default == BackgroundTaskHost.exe

Background Task Type Name

Unregistering a Background Task

```
foreach (var task in BackgroundTaskRegistration.AllTasks)
{
    if (task.Value.Name == "TimeZoneTask")
        task.Value.Unregister(false);
}
```



*Do not cancel instances of this
task that are currently executing*

DEMO

Implementing a Background Task

Triggers

- Background tasks run when triggers fire
 - And when conditions are satisfied (more later)
- A background task must have exactly one trigger
 - Specified with `BackgroundTaskBuilder.SetTrigger`
- Some triggers can be used by any app
 - `SystemTrigger`, `MaintenanceTrigger`, and `LocationTrigger`
- Others can only be used by lock-screen apps
 - `TimeTrigger`, `PushNotificationTrigger`, and `ControlChannelTrigger`

Trigger Types

Type	Availability	Process	Description
SystemTrigger	Depends on trigger type	BTH	See System Trigger Types
MaintenanceTrigger	All apps	BTH	Executes task periodically at most every 15 minutes; requires AC power
LocationTrigger (8.1)	All apps	BTH	Executes task when specified location criteria are satisfied (geofencing)
TimeTrigger	Lock Screen	BTH	Like MaintenanceTrigger, but works on battery power, too
PushNotificationTrigger	Lock Screen	BTH or App	Executes task when raw notification is received from Windows Notification Service
ControlChannelTrigger	Lock Screen	App	For apps that require real-time communications; not available to JavaScript apps; use is discouraged

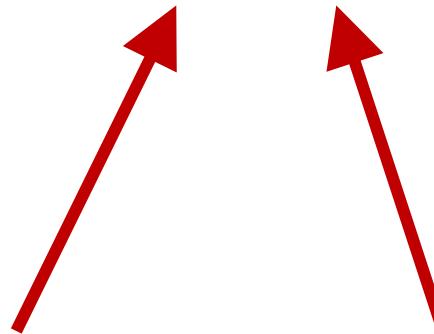
* Windows 8.1 also introduces DeviceUseTrigger and DeviceServicingTrigger (not shown)

System Trigger Types for All Apps

Type	Description
InternetAvailable	Internet becomes available
LockScreenApplicationAdded	Your app is added to the lock screen
LockScreenApplicationRemoved	Your app is removed from the lock screen
NetworkStateChange	Network change occurs (e.g., cost change)
OnlineIdConnectedStateChange	Online ID associated with the account changes
ServicingComplete	Your app is updated
SmsReceived	Device receives an SMS (text) message
TimeZoneChange	Device's time zone changes
BackgroundWorkCostChange (8.1)	Background work cost changes
NfcStateChange (8.1)	Device's NFC state changes (e.g., device arrived)

Registering a Periodic Task

```
// Register PeriodicTask to execute every 15 minutes
var builder = new BackgroundTaskBuilder();
builder.Name = "PeriodicTask";
builder.TaskEntryPoint = "SampleBackgroundTask.PeriodicTask";
builder.SetTrigger(new MaintenanceTrigger(15, false));
builder.Register();
```



Minimum 15 minutes

Recurring rather than one-shot

Conditions

- Background tasks can have conditions attached
 - Task runs when trigger fires and all conditions are met
 - Added with BackgroundTaskBuilder.AddCondition
- Windows.ApplicationModel.Background.SystemCondition class models system conditions
 - Whether Internet is available
 - Whether user is logged in
 - Whether user is present

Condition Types

Type	Description
InternetAvailable	Runs background task if Internet is available
InternetNotAvailable	Runs background task if Internet is not available
SessionConnected	Runs background task if user is logged in
SessionDisconnected	Runs background task if user is not logged in
UserNotPresent	Runs background task if user is not present
UserPresent	Runs background task if user is present
FreeNetworkAvailable (8.1)	Runs background task if free network is available
BackgroundWorkCostNotHigh (8.1)	Runs background task if cost is low

Adding a Condition

```
var builder = new BackgroundTaskBuilder();
builder.Name = "TimeZoneTask";
builder.TaskEntryPoint = "SampleBackgroundTask.TimeZoneTask";

builder.SetTrigger(
    new SystemTrigger(SystemTriggerType.TimeZoneChange, false)
);

builder.AddCondition(
    new SystemCondition(SystemConditionType.InternetAvailable)
);

builder.Register();
```

DEMO

Periodic Tasks

Background File Transfers

- Transfer data while app is running or suspended
 - BackgroundDownloader and BackgroundUploader classes
 - Primarily for large transfers (video, music, large images)
 - Transfers continue through user log off/on, reboots, and network switches
 - Downloads resume; uploads restart
- Features
 - Supports HTTP, HTTPS, and FTP
 - Supports pause, resume, cancel, and read
 - Windows 8.1 adds support for group transfers via BackgroundTransferGroup
 - Windows 8.1 also makes it easy to update tiles and provide toast notifications regarding download progress

Performing a Background Download

```
void StartDownload(Uri uri, IStorageFile file, CancellationToken ct)
{
    var bd = new BackgroundDownloader();
    DownloadOperation dop = bd.CreateDownload(uri, file);
    var p = new Progress<DownloadOperation>(DownloadProgress);
    dop.StartAsync().AsTask(ct, p).ContinueWith(DownloadDone);
}

void DownloadProgress(DownloadOperation dop)
{
    var percent = (dop.Progress.BytesReceived * 100) /
        dop.Progress.TotalBytesToReceive;
}

void DownloadDone(Task<DownloadOperation> dop,
    DownloadOperation dopIfCanceled)
{
    // Complete!
}
```

Checking Completion Status

```
void DownloadDone(Task<DownloadOperation> task,
                  DownloadOperation dopIfCanceled)
{
    try
    {
        // If this doesn't throw, download completed successfully
        var dop = await task;
    }
    catch (OperationCanceledException)
    {
        // Download canceled
    }
    catch (Exception ex)
    {
        // Handle other errors here
        WebErrorStatus status =
            BackgroundTransferError.GetStatus(ex.HResult);
    }
}
```

Reattaching when the App Restarts

```
async void Reattach(CancellationToken ct)
{
    var p = new Progress<DownloadOperation>(DownloadProgress);

    IReadOnlyList<DownloadOperation> dops =
        await BackgroundDownloader.GetCurrentDownloadsAsync();

    foreach (var dop in dops)
    {
        dop.AttachAsync()
            .AsTask(ct.Token, p)
            .ContinueWith(t => DownloadDone(t, dop));
    }
}
```

DEMO

Background File Transfers

Download the Code

- Windows 8.1 samples

<http://1drv.ms/1n0JLwN>

- Universal App Samples

<http://1drv.ms/1ij2K71>