Elements of a Windows tailored app: The Developer Story

M3 (PDC-4) Edition

This version targets the PDC-4 build and is the final installment.

*For questions please contact **manavm** and **ialegrow***

*To submit feedback or corrections on this document please send mail to **win8eom***

*For questions and how-to's on developing modern applications please send mail to **dog8appbld***

1 TABLE OF CONTENTS

2 OVERVIEW

Windows 8 introduces a new programming paradigm that developers can use to create a modern, polished user experience. This new, Modern Application model makes it simple to get apps on the machine and makes it just as simple to remove them. Modern apps push application experiences to the forefront, immersing users in the experiences they offer.

This document, written for developers, describes the core elements of the Modern Application model and all the capabilities it offers developers. It provides an overview of the Modern Application model, then walks through the application development lifecycle, from application development and packaging, to Modern Shell (MoSh) integration and controls, to activation, to integration with the App Store.

2.1 WINDOWS 8 – A NEW APPLICATION PARADIGM

A key benefit provided by Windows 8 is the availability of a new programming paradigm. You can use this new programming paradigm to develop applications with an enhanced look-and-feel that provide a better user experience. There are three key elements of this new paradigm:

- **A new app model.** Windows 8 ships with a new application model that will earn consumer confidence and trust by making it safe and simple to get apps on the machine and just as safe and simple to remove them. The new application model gives developers an additional language choice: HTML5, now a first-class Windows app development language. It also provides the Windows Runtime, a set of API that exposes the power of Windows in several languages.
- **A new, immersive shell.** A new modern shell, the MoSh, is the place where Windows 8 app model applications are experienced. Rather than highlighting Windows UX, the MoSh pushes application experiences to the forefront, immersing users in the experiences they offer. The modern shell is optimized for touch and includes simplified window management, notifications, process lifetime management, and more. A new set of controls, animations, patterns and personality characteristics showcase a modern, polished experience.
- **An application store / marketplace**. The Windows organization is building a marketplace where consumers obtain modern apps and developers make money building Windows apps.

2.2 WHAT IS A MODERN APP?

A Modern Application is an app that runs only in the new immersive shell (the MoSh). A modern app has the following characteristics:

- It is written to take advantage of the immersive experience in the new shell.
    - It provides a UI / look & feel that is compliant with the MoSh UX model.
    - It supports modern activation, windowing, and process lifetime management.

    - It supports app contracts. App contracts define requirements that the application must follow to participate in Windows controlled UI flows and experiences, such as the following:
        - Tiles, docking, splash screen, search, share, toasts, push notifications, file picking, app settings, connect
- It is delivered through the new deployment technology (AppX).
- It is installed per user.
- It installs and uninstalls cleanly.
- It doesn’t change the state of the OS in an irreversible way.
- It runs in an Application Container.
- It can be terminated predictably by the user.

2.3 WHY DEVELOPERS WILL WANT TO WRITE MODERN APPS

The Modern Application model—along with a set of development tools, a developer center, and an app store—enables you to reduce development costs for building modern applications that leverage the power of Windows and make more money.

The Modern Application model provides the following benefits:

- **Quick Development:** The Developer Center, new tools optimized for modern apps, multiple language choices, and robust presentation frameworks help you quickly develop full-functioning apps that look great.
- **Quick Deployment**: The App Store makes deployment simple.
- **Quick Updates:** The App Store enables seamless app updates.
- **Quick Money:** The new app model and platform provide system protection, user data and privacy protection, and a restricted, partitioned execution environment. These benefits, along with confidence inspired by the App Store validation, reputation information, and one-stop app updating encourage more application installs and generate more money for you.

The Windows 8 developer experience starts at the new Developer Center, which provides streamlined access to state of the art tools and community information about developing apps for Windows 8. The new tools and underlying platform enable the developer to choose a language and presentation framework—HTML/JS/CSS, XAML/C++, or XAML/c#—and quickly get started. With the availability of HTML/JS as a Windows development option, web develops can use their existing skills to begin creating powerful client applications.

The new application model makes it easy for you to create applications that quickly install and uninstall without changing the state of the system. The protected execution environment makes it easier for users to trust that your application won’t harm their system or invade their privacy. The new developer tools make it easy for you to design your app’s UI, perform runtime debugging, and publish to the App

Store. The developer tools even check that your app meets store requirements as you're coding it! The developer tools even check that your app meets store requirements as you code, ensuring that you'll be ready to sell your app as soon as it's finished.

Once your app is in the App Store, it will be matched up with customers based on curated lists, recommendations, and reviews. Updating your app is easy: just submit the update to the store, and it will seamlessly deliver the update to your customers.

More app downloads means more money for you. The safety of the new platform instills customer confidence, and the ability to simply click to install and run new apps encourages customers to download more apps. Each month, the Windows store will send you a check and provide access to telemetry for your apps.

2.4 STEPS TO BUILDING A MODERN APP

Creating a modern application involves five steps:

1. **Author the app's core functionality.** If you're building a photo viewing app or a music player, build that functionality first, keeping in mind the look and feel and UX guidelines of modern applications (such as. touch optimization, a full-screen experience, and so on).
2. **Tailor the app to Windows, the form factor, and the experience.** Implement the required and recommended set of application contracts. These contracts define requirements that applications must follow to participate in Windows Controlled UI Flows and Experiences. Further contract specific details are outlined later in the document.
3. **Package the app.** Package the application using the modern packaging technology. Visual Studio does this work for you.
4. **Test and Debug the app.** Ensure the app is working as expected.
5. **Upload the app to the App Store.** Modern applications are installed by users from the App Store. Again, Visual Studio assists with this process, though there's an online portal available as well.

2.4.1 CHOOSING TO BUILD A WWA OR AN MCA IMMERSIVE APPLICATION

There are two types of application technologies available in Windows 8 for creating immersive applications (internally and externally), Windows Web Applications (WWAs) that are written in HTML, CSS, and JavaScript, and Modern Client Applications (MCAs) that are written in C++ or C# using the WinRT UI framework built into Windows 8 (formerly known as Jupiter).

Windows 8 Immersive Apps	**User Interface (View)**	**Application Logic**

are:

- AppX Packaged
- Installed through the App Store
- Run in AppContainer

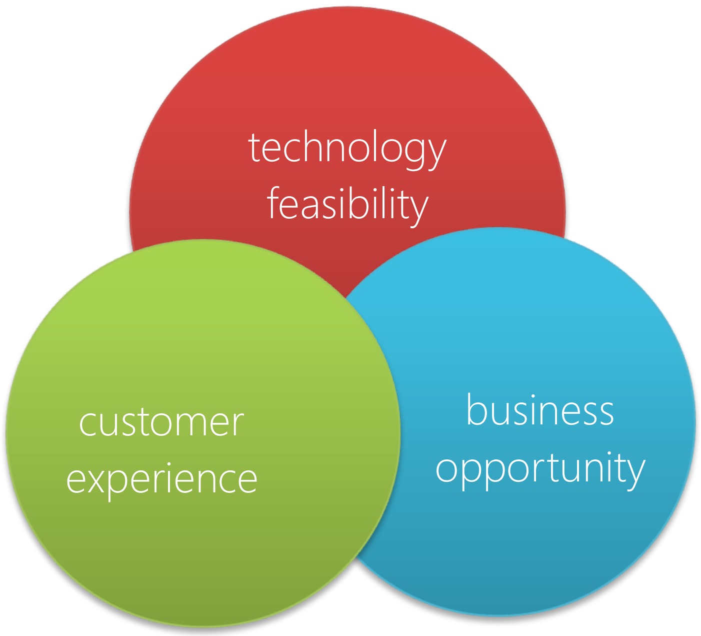
Technology for Immersive Apps	(Model/ViewModel/Controller) Technology for Immersive Apps
HTML + JavaScript W3C®	• JavaScript • C++ • C#
WinRT (Windows Runtime) UI Wind8ws	• C++ • C#

- Both WWA and MCA applications, through HTML/CSS/JavaScript and through Win RT UI respectively, can use all of the standard Windows 8 UI controls (timelines for availability may differ between WWA and MCA applications). These controls render with the with the Windows 8 look and feel.
- **Both** WWA and MCA applications have access to the WinRT APIs and are enabled through inbox libraries and components, not requiring external dependencies on Framework Packages.
- **Both** WWA and MCA applications can be extremely responsive for touch interaction and can have great performance on low and high end hardware, especially when using the WinRT APIs.
- **Both** WWA and MCA applications have support through Visual Studio for Windows 8 and tools like the Expression Suite of application design tools (timelines for availability of various feature support may differ between WWA and MCA applications).
- **Both** WWA and MCA applications can reuse existing web content (MCAs through hosting Trident) and native code (WWAs through WinRT Native Extensibility) as a part of their application logic.

Scenario Focused Development

When developing software or technologies, it's common to focus on technical considerations, such as architectures, platforms, and systems, rather than the experiences and scenarios the software should support.

The Windows 8 application model is designed so that you can focus on experiences and scenarios: in fact, scenario focused engineering is at the heart of Windows 8 and the immersive application model. No matter which application programming technology you choose—a WWA coded with HTML and JavaScript, an MCA that uses C++ or C#, or a combination of the two—you can build a beautiful application experience that conforms to the look and feel of Windows 8's immersive shell. The following are three pivots that you can use to make a choice between a WWA and an MCA.



The following decision matrices and information should help you decide the best technology for your applications.

Customer Experiences:

Experiences	WWA	MCA
Paint applications	Yes. Consider using the HTML 5 Canvas that allows you to draw pixels!	Yes. DirectX provides rich features for textures, pixel shaders, etc.
RSS Readers/Web Reader Applications	Yes.	Yes.
Sprite Based Games	Yes.	Yes.
3-D Games	No. Even though you can build applications that are pseudo 3-D using Canvas (http://dev.opera.com/articles/view/creating-pseudo-3d-games-with-html-5-can-1/), this will not have the level of performance of an	Yes.

	MCA	
e-Book Reader Applications	Yes.	Yes.
Photography Editing Applications	Yes. Many photo effects can be processed off-screen through native code using WinRT.	Yes.
Travel Applications	Yes.	Yes. WinRT uses XAML, but this XAML is not compatible with Silverlight. When hosting things like the Bing Maps control, MCAs will need to host Trident like a WWA manually or consider using alternative hosting mechanism for Silverlight and Flash which may not run in the Application Container.
Music Applications	Yes.	Yes.

Business Opportunities:

Experiences	WWA	MCA
Would like to re-use large portions of an existing web property so that Windows 8 immersive application updates along with an externally facing web site	Yes.	No. Even though it is possible through hosting Trident, the overhead and complexity of doing this doesn't justify using an MCA over a WWA when the WWA hosts Trident for you.
Share large portions of native code (backend code, not UI code) with my desktop applications	Yes. You will have to expose native code through WinRT which may change your data model significantly.	Yes.
Build a set of common HTML resources to use across my sites and applications	Yes.	No.
Build a set of common XAML resources across Windows Phone, Windows 8, and Xbox	No.	Yes. This is a huge advantage for developers building applications using XAML for Windows Phone and Xbox 360.

Host web based ads to present to users as a form of monetization	Yes. There is amazing work for security here with web and local compartments that prevent ads from accessing WinRT.	No. Many web based ad properties require the logic of a site and URLs. It is possible to host web based ads by hosting Trident, but this work would add unneeded complexity when writing a WWA would provide this more easily.

Technology Feasibility:

Experiences	WWA	MCA
Direct Access to the GPU through DirectX, DirectImage, DirectComposite, D2D, D3D, and so on.	No.	Yes.
Use of the GPU through Blobs and Events behind the scenes without drawing (DirectCompute)	Yes.	Yes.
Use of Frameworks like Adobe AIR and XNA	No.	No.
Developers with XAML or native code experience	No.	Yes.
Designers with HTML, JavaScript, Flash Experience	Yes.	No.
Designers with XAML, DUI, Splash UI, Net UI experience	No.	Yes.
Must guarantee uninterrupted high performance with advanced multi-threading for intensive background operations	Yes. Though JavaScript is traditionally single-threaded (run on the UI thread), most WinRT APIs are asynchronous. Additionally, WWAs support "HTML5" web workers.	Yes.

2.5 TAXONOMY

Taxonomy	
MoGo, Start	The new launcher for Windows 8. It is intended to be the replacement for the start menu. It includes both MoSh and classic apps.
Edgy	The collection of edge-based UI in the immersive shell: switching apps, the Charms Bar, and the App Bar.

Swipe	A touch gesture that invokes edge-based UI. The gesture starts at the edge of the screen and then drags perpendicular to the edge and interior to the screen.
Snapped App	When two applications are displayed at one time side by side, there will be one app which is smaller and one that is larger. The snapped application is the smaller application.
MoBody	The portion of the screen where the main app runs. In full-screen mode, there is no snapped app shown and the MoBody app occupies the entire screen. When a snapped app is present, the MoBody resizes itself so that it occupies the remainder of the screen.
Charms Bar	Formerly known as “the lucky charms bar,” a vertical bar containing entry points to a number of different system UI components. It is a component of Edgy.
App Bars	Modern application-defined UI affordances that typically provide access to application commands, and are invoked with a vertical swipe. They are invoked through Edgy.
App Switching	An edgy gesture from the left side of the screen to switch between your most recently used apps.
MoSh Application	A packaged application that runs in MoSh—it runs per user and in full-screen mode, has a manifest with declared capabilities, has stored state, and runs in an App Container.
Classic Application	A Windows application as it is in the current Windows 7 world. These applications are installed per machine rather than per user, do not have declarative install, and do not run in an Application Container.
WWA	Windows Web Application – a “modern application” written in client-side web technologies: JavaScript, HTML, CSS, and SVG. These applications don’t run in a web browser; rather, they run inside the WWA host, a specialized execution environment for these WWA applications.
MCA	Modern Client Application – a “modern application” written in a traditional Windows programming language: C, C++, C#, VB.
Manifest	A required file that is part of the package that defines several portions of the application, such as its identity, metadata, extensions, and capabilities.
WinRT, aka Windows RunTime aka REX APIs	The set of new Windows APIs that will simplify Windows application development. These APIs are projected into all of the Windows 8 supported languages (including JavaScript).
Application Container (lowbox)	A new security context within which Modern Applications are executed. Application Containers help protect users from attack by running with greatly restricted capabilities on Windows 8. While Application Containers do not protect against all forms of attack, they significantly reduce the ability of an attack to install malicious code or compromise user’s data and

	privacy.
Package	A type of zip file that contains the application: it contains the files necessary for deployment, including resources and a manifest file.
Capability	A desired behavior declared in the package manifest that describes a characteristic of the application in terms of custom scenarios. For example, an application's ability to access a user's photo library is a capability.
Full Trust Application	An application that is running in the same way as applications run in Windows 7: with the full privileges of the user. Full trust applications are not permitted in the immersive shell.
Modern API Component	A set of programming interfaces available in the developer platform that expose, enable, or participate in the delivery of capabilities to applications. As described in the Windows Runtime, modern runtime API components are implemented as a CoClass. Each component may expose functionality with varying levels of capabilities and risk.
Hardened Component	A component able to handle untrusted data it receives from untrusted code. As part of the hardening process, the component goes through a thorough security review with the Secure Windows Initiative Team (SWI).
Broker	The Windows 8 confidence model depends on components that act as a "go between" or "bridge" between processes of differing trust levels. In particular, this term is used to describe components that offer functionality through another process (broker process) to enable capabilities that are otherwise not directly accessible from the Application Container environment.

3 MODERN APP MODEL

With Windows 8, Microsoft introduces the latest evolution of applications: Modern Applications. As described earlier, Modern Applications run per user, have declared capabilities, and execute in Application Containers. Modern Applications can be written in C, C++, C#, JavaScript and can use WinUI ("native XAML") or the WWA Host (augmented HTML5) for their presentation model.

Modern Applications run only in the new Modern Shell (MoSh). Modern Applications are acquired only via the App Store, which provides modern app developers with a marketplace where they can make money and provides an easy way to deploy, service, and update application.

3.1 SUMMARY OF MODERN APPLICATION PROPERTIES

Modern Applications have the following characteristics:

1. They are written for the immersive experience of the modern shell.
2. They are acquired through the App Store.
3. They are delivered through new packaging and deployment technology.
4. They are signed.
5. They are installed per user.
6. They install and uninstall cleanly.
7. They don't change the state of the OS in an irreversible way.
8. They run in Application Containers (formerly LowBox).
9. They have clearly separated and isolated application state.
10. They can always be predictably terminated by the user.

Modern Applications can be further classified as Modern Client Applications (**MCAs**) and Windows Web Applications (**WWAs**). MCAs are packaged applications written in supported Windows languages, such as C, C#, and C++. The common UI framework choice for these apps is the Windows Runtime UI (formerly called Jupiter), though others are available for use, such as DirectX. These applications can call Win32 APIs in addition to WinRT APIs, though there are some restrictions around this API surface imposed by the Application Container, a restricted execution environment where modern apps run with a reduced subset of the user's privileges, and the Modern SDK (approved subset of Win32 APIs that may be called by modern applications).

WWAs are packaged applications written in standard client-side web technologies: JavaScript, HTML, CSS, and SVG. To help create a strong WWA experience for both end-users and developers, Windows and DevDiv are making heavy investments in the web platform by creating a new set of UI controls for WWAs, improving touch interaction, and delivering robust tooling. Like their Modern Client Applications counterparts, WWAs run only in the Immersive Shell. They can include native code, but only APIs available via WinRT. WWAs only run in Application Containers.

3.2 PACKAGING & MANIFEST

Modern Applications are acquired from the App Store in the form of **packages**. A package is the smallest atomic unit of deployment, management, and servicing of software in the Windows 8 application model. A package is essentially an OPC-based ZIP container that includes a package manifest and a set of binaries. Packages have the file extension ".appx". There are two types of packages: application packages and framework packages. These are described in the sections that follow..

When creating a Modern Application, you (as the application developer) don't write install and uninstall routines. Rather, you author an XML manifest, known as a package manifest, which describes the application's structure and capabilities to the system. This package manifest is included in the .appx package when you upload the application to the App Store. Windows is then responsible for acting upon the information in the manifest when installing and uninstalling the application.

3.2.1 A SAMPLE MANIFEST

The first section of the manifest references the manifest schema and declares the encoding, which is always UTF-8 in Windows 8. Manifest validation is provided by an XSD, which will be available externally and with accompanying tool support at RTM. It is available for internal use as part of the builds: \\winbuilds\release\WinMain\<build>\x86fre\bin\AppxTools\MakeAppx.

```
<?xml version="1.0" encoding="utf-8"?>

<Package xmlns="http://schemas.microsoft.com/appx/2010/manifest">
```

A package contains a single manifest. The package is identified using a unique identifier, known as the **PackageID**, which is a tuple made up of the following attributes: **name**, **publisher**, **processorArchitecture**, **version**, and a **resourceID**. These items must be specified in the manifest.

3.2.1.1 IDENTITY

```
  <!--IDENTITY-->
  <Identity Name="contoso.photoviewer"
          ProcessorArchitecture="x86"
          Publisher="CN=Contoso,O=Contoso Inc.,L=Redmond
,S=Washington,C=US"
          Version="2.5.1.6"
          ResourceId="en-us" />
```

3.2.1.2 PREREQUISITES

Packages will often require some minimum level of prerequisite functionality be resident on client machines, as defined by the operating system and software requirements. These are specified in the **Prerequisites** section.

```
<!--PREREQUISITES-->
<Prerequisites>
  <OSMinVersion>6.2</OSMinVersion>
  <OSMaxVersionTested>6.2</OSMaxVersionTested>
</Prerequisites>
```

3.2.1.3 PROPERTIES

The **Properties** section contains additional data about the software package, such as the package's type, classification, and policies.

```
<!--PROPERTIES-->
<Properties>
  <Framework>false</Framework>
  <DisplayName>Contoso Photo Viewer</DisplayName>
  <PublisherDisplayName>Contoso INC</PublisherDisplayName>
  <Description>View and edit your photos</Description>
  <Logo>foo.ico</Logo>
</Properties>
```

3.2.1.4 RESOURCES

The **Resources** section describes the UI resources provided by the application. This section declares the languages that the package supports. Every package manifest must contain a **Resources** section.

```
<!--RESOURCES-->
<Resources>
  <Resource Language="en-us"/>
  <Resource Language="fr-fr"/>
</Resources>
```

3.2.1.5 DEPENDENCIES

Dependencies between packages must be explicitly declared so that Windows can check and appropriately resolve the dependencies during application acquisition. The package that takes a dependency on another package is referred to as the **dependent package**. The package that fulfills the dependency is referred to as the **dependency package**. When specifying a dependency, you must specify the publisher.

The following limitations are enforced for dependencies:

- Packages can only declare a dependency on Framework Packages (framework = true in the Package Manifest).
- Packages cannot declare a dependency on themselves.
- Framework Packages cannot declare a dependency on other packages (Framework packages with declared dependencies are just ignored).
- Packages with more than 16 dependencies declared will fail.
- Packages cannot declare dependencies on packages with dependencies. There is no support for anything except direct dependencies. This means that the dependency graph will never be more than one level deep.

```
  <!--DEPENDENCIES-->
  <Dependencies>
    <Dependency Name="Microsoft.PlatformExtensions.Corsica"
      Publisher="CN=Microsoft Corporation, O=Microsoft Inc., L=Redmond, 
S=Washington, C=US"
      MinVersion="4.5.0.0"/>
  </Dependencies>
```

3.2.1.6 CAPABILITIES

Capabilities are declared to indicate that an application requires additional functionality beyond what is provided to an application by default. For access to a user's library folders or the network, for instance, the corresponding capability must be specified.

```
  <!--CAPABILITIES-->
  <Capabilities>
    <Capability Name="picturesLibrary"/>
    <Capability Name="videosLibrary"/>
    <Capability Name="musicLibrary"/>
    <DeviceCapability Name="webcam"/>
    <DeviceCapability Name="A3A8783C-4844-4b09-9E7A-9E8003D4E20B"/>
  </Capabilities>
```

3.2.1.7 APPLICATIONS

A package may contain one or more applications. A package's applications are specified in the **Applications** section of the package manifest. Note that these applications cannot be installed separately—the package itself is the smallest unit of software that can be added, updated, or removed from the system.

```
<!--APPLICATIONS-->
<Applications>
<!--Web app-->
<Application Id="Office.PowerPoint" StartPage="PowerPoint\Index.html">
…
</Application>

<!--Native app-->
<Application Id="Office.Word" Executable="Work\WinWord.exe"
EntryPoint="Office.Winword.Class" >
…
</Application>
```

3.2.1.8 VISUAL ELEMENTS

As the name suggests, the **Visual Elements** section is used to declare the visual aspects of the application. These aspects include display name, application logos that are displayed in multiple places on the system, and others. The **Visual Elements** tag is contained within (a child of) the **Application** tag within the manifest.

- **DisplayName** – Application DisplayName is a required attribute that defines the default name for the application as the developer intends must appear on the app. DisplayName is localizable.
- **Logo –** Application Logo is a required attribute that supplies the common image, used for the MoGo 1x1 tile. Also used in File Extension Logo unless replaced by developer. Logo is localizable. Required sizes are 1x@152x152. Supported formats are .png and .jpg.
- **SmallLogo –** Application SmallLogo is also a required attribute that supplies the common image, used for Tiles (2x1 and 1x1), MoGo - All Programs view, MoSettings. Also used in File Extension Logo unless replaced by developer. SmallLogo is localizable. Required sizes are 1x@32x32. Supported formats are .png and .jpg.
- **Description –** A string for the description of the application. This string is localizable. This is also a required attribute. All per-application extensions can use the value of this attribute if they require it.
- **ForegroundText –** Specifies a tile's foreground text is dark or light. This is also a required attribute.
- **BackgroundColor –** Specifies a tile's background color. This is also a required attribute.
- **ToastCapable –** This is an optional Boolean attribute. It includes information enabling application to show or not show toast.
- **InitialRotationPreference –** This is an optional attribute. It's value is used to set the rotation preference on behalf of the app so that when the app is displayed it will have that setting. If developers want a portrait-only app they can just set this once in the manifest. It is possible for apps to change this preference at runtime via Rotation Manager APIs. It also provides the rotation preference for the app splash screen.
- **DefaultTile –** This optional element contains the following attributes;
    - **ShortName:** Optional. Name appears on the tile. Allowed limit is 13 characters only.

    - **ShowName:** Optional. Specifies whether the app name is displayed by default when the tile changes from the default state
    - **WideLogo:** Optional. Used for MoSettings. Required size is 1x@312x152. Allowed file formats are .jpeg, .jpg and .png.
- **LockScreen –** This optional element contains the following attributes;
    - **Notification:** Required. Can be set to one of the 3 options; tileText, badge, badgeAndTileText.
    - **BadgeLogo:** Optional. Used to provide the logo for the Lock Screen, next to badge notifications eg: Mail Icon. Note that this image is required if Notification is set to badge or badgeAndTileText. Required size is 1x@24x24. Allowed file formats are .jpeg, .jpg and .png. Image must be monochromatic (white & transparent pixels only)
- **SplashScreen** – This required element contains the following attributes;
    - **BackgroundColor: Optional.** Specifies the splash screen's background color. If not specified, default background color will be used.
    - **Image: Required.** Specifies the path to the image required for splash screen. Required size is 1x@624x304. Allowed file formats are .jpeg, .jpg and .png.

```
<!—VISUAL ELEMENTS-->
<VisualElements DisplayName="Microsoft Word" Logo="Images\word.png"
SmallLogo="Images\word2.png" Description="Word from Microsoft Office
family" ForegroundText="dark" BackgroundColor="#777777"
ToastCapable="true" InitialRotationPreference="portraitAndFlipped">
        <DefaultTile ShortName="Word" ShowName="true" WideLogo="bar.png"/>
        <LockScreen Notification="badgeAndTileText"
BadgeLogo="Icons\lockscreenimage.png"/>
        <SplashScreen BackgroundColor="blue" Image="photo.jpg"/>
</VisualElements>
```

3.2.1.9 APPLICATION CONTENT URI RULES

ApplicationContentUriRules are a set of Windows Web Application specific declarations.

```
  <!—APPLICATION CONTENT URI RULES-->
  <ApplicationContentUriRules>
        <Rule Type="include" Match="http://www.example.com/"/>
        <Rule Type="exclude" Match="http://www.w3.org/People/Dürst/"/>
        <Rule Type="exclude" Match="*.pdf"/>
   </ApplicationContentUriRules>
```

3.2.1.10 EXTENSION POINTS

An **extension point** (EP) is a location defined by an application or system that permits another application to register to have code executed or resources used. Examples include file type associations, share target, file picker and others. Extension points are a mechanism by which an application can add functionality in a manner defined by the operating system thereby providing a deeper, richer, and more connected experience.

There are two distinct types of extensions:

- Per application extensions, that apply only to a single application
- Per package extensions, that apply to all applications in a package

Per-Application Extensions are scoped to a particular application and encapsulated within an application element in the package manifest. e.g. file type extensions, sharing extensions, etc. Extensions are category specific.

Extensions

Name	**Description**
fileTypeAssociation	A supported FileType handled by the information declared in the extension (icon, ACID, etc.)
protocol	Defines the protocol to be taken over by the application.
autoPlayContent	Provides support for AutoPlay in the Mosh.
autoPlayDevice	Provides support for AutoPlay in the Mosh.
shareTarget	Developers of sharing applications can utilize the share functionality so that users can share to their application from any source application. Eg. Facebook that wants to provide share service.
sendTarget	Developers of device applications can utilize the SendTo functionality so that users can send to their application from any source application. We call the apps that receive SendTo-able content and make it visible to other users target applications. The target application is displayed in a flyout window. An example target application is Windows Phone application.
search	This extension is declared by an app that wants to provide search service to other apps. For example, a bing app manifest would declare this extension, and bing would come up as an option for searching from other apps.
filePicker	The application has the ability to be hosted in the Picker and provide items, specifically file-like items to be picked by the user, e.g. a photo from their Flickr account.
backgroundTasks	Declare background tasks of app such as network access, allow download, etc. when suspended
contactPicker	Allow applications that have a notion of people to provide contact data to other applications. Scenario: Andy is using Bing Maps, and clicks on the “find a friend” button. This opens the People Picker, where Andy selects his friend Bob. Andy then closes the Picker, and Bing Maps shows Bob’s house on the map.
userTileProvider	Applications that support User Tile Provider functionality will be capable of

	providing a set of assets used to represent the user throughout the system and in third party applications. Users will launch the Modern Settings app and be provided with a list of User Tile provider apps that they can launch in User Tile mode.
cameraSettings	Webcam manufacturers can build control applets for their devices that expose fun effects such as various image filters, stars swirling around a user's head etc. Webcam capture applications and video chat applications, can leverage these applets to offer an enhanced experience to the user.
printTaskSettings	Enable print manufacturers to replace the basic print settings experience with a Device Companion app, that allows IHVs to showcase all the features and functionalities of the printer.

```
<!—APPLICATION EXTENSIONS-->
   <Extensions>
       <Extension Category="windows.fileTypeAssociation">
         <FileTypeAssociation Name="documenttypes">
           <InfoTip>Microsoft Office openable types</InfoTip>
           <EditFlags OpenIsSafe="true" AlwaysUnsafe="false"/>
           <SupportedFileTypes>
             <FileType MediaType="application/msword">.doc</FileType>
             <FileType>.docx</FileType>
           </SupportedFileTypes>
         </FileTypeAssociation>
       </Extension>

       <Extension Category="windows.protocol">
         <Protocol Name="mailto"/>
       </Extension>

       <Extension Category="windows.autoPlayContent"
EntryPoint="Moustache.View">
         <AutoPlayContent>
           <LaunchAction ActivationContext="moustacheContent"
ActionDisplayName="Funny Moustaches"
ContentEvent="ShowPicturesOnArrival"/>
         </AutoPlayContent>
       </Extension>

       <Extension Category="windows.autoPlayDevice"
EntryPoint="Moustache.Sync">
         <AutoPlayDevice>
           <LaunchAction ActivationContext="cameraDevice"
ActionDisplayName="Funny Moustaches" DeviceEvent="imageSource"/>
         </AutoPlayDevice>
       </Extension>

       <Extension Category="windows.shareTarget"
EntryPoint="MyActivateableClassId.baz">
         <ShareTarget>
```

```
            <SupportedFileTypes SupportsAnyFileType="false">
              <FileType>.pdf</FileType>
              <FileType>.customFileType</FileType>
            </SupportedFileTypes>
            <DataPackageFormat>text</DataPackageFormat>
            <DataPackageFormat>HTML 2.0 Format</DataPackageFormat>
          </ShareTarget>
        </Extension>

        <Extension Category="windows.search"
EntryPoint="MyActivateableClassId.baz"/>

        <Extension Category="windows.filePicker"
EntryPoint="MyActivateableClassId.baz">
          <FilePicker>
            <SupportedFileTypes SupportsAnyFileType="false">
              <FileType>.png</FileType>
              <FileType>.bmp</FileType>
            </SupportedFileTypes>
          </FilePicker>
        </Extension>

        <Extension Category="windows.sendTarget"
EntryPoint="MyActivateableClassId.baz">
          <SendTarget>
            <SupportedFileTypes SupportsAnyFileType="false">
              <FileType>.xps</FileType>
              <FileType>.customFileType</FileType>
            </SupportedFileTypes>
            <DataPackageFormat>text</DataPackageFormat>
            <DataPackageFormat>HTML 2.0 Format</DataPackageFormat>
          </SendTarget>
        </Extension>

        <Extension Category="windows.contactPicker"
EntryPoint="MyActivateableClassId.baz"/>

        <Extension Category="windows.backgroundTasks"
EntryPoint="Fabrikam.BackgroundTask" Executable="Winword.background.exe">
          <BackgroundTasks>
            <Task Type="audio"/>
            <Task Type="network"/>
            <Task Type="timer"/>
          </BackgroundTasks>
        </Extension>

      </Extensions>
```

Package-specific extensions are specified by the **Extensions** element, a child of the **Package** element. Each application included in the package may also include app-specific extensions, defined later in this document.

Extensions

Name	**Description**
InProcessServer	Allows for registration of 3rd party components with Windows Runtime. There are two types of activatable classes: In process activatable classes which are implemented as DLLs and out of process activatable classes implemented as EXE servers. Each type of activatable classes have distinct registration requirements
OutOfProcessServer	
ProxyStub	Interface proxy registration is necessary for each interface that is marshaled across context boundaries.
GameExplorer	Game developers will include a relative path to a Games Definition File (GDF) along with icon or thumbnail files for Parental Controls.
Certificates	Enable rising star developers to install and use per-app certificates.
DigitalRightsManagement	Required for usage by the PlayReady DRM framework package only.

```
<!--PER-PACKAGE EXTENSIONS-->
  <Extensions>
    <Extension Category="windows.activatableClass.inProcessServer">
      <InProcessServer>
        <Path>Fabrikam.dll</Path>
        <ActivatableClass ActivatableClassId="Fabrikam.Foo"
ThreadingModel="STA">
          <ActivatableClassAttribute Name="AssemblyName" Type="string"
Value="Fabrikam.dll"/>
          <ActivatableClassAttribute Name="MajorVersion" Type="integer"
Value="4"/>
          <ActivatableClassAttribute Name="MinorVersion" Type="integer"
Value="5"/>
        </ActivatableClass>
        <ActivatableClass ActivatableClassId="Fabrikam.Bar"
ThreadingModel="both"/>
      </InProcessServer>
    </Extension>

    <Extension Category="windows.activatableClass.outOfProcessServer">
      <OutOfProcessServer ServerName="WinwordBackgroundTasks">
        <Path>Winword.background.exe</Path>
        <Arguments>/i winword.config</Arguments>
        <Instancing>singleInstance</Instancing>
        <ActivatableClass ActivatableClassId="Winword.Baz"/>
      </OutOfProcessServer>
    </Extension>

    <Extension Category="windows.activatableClass.proxyStub">
      <ProxyStub ClassId="12345678-1234-1234-1234-123456789ABC">
        <Path>ProxyStub.dll</Path>
        <Interface Name="Fabrikam.IBaz" InterfaceId="12345678-1234-1234-
1234-123456789ABC"/>
        <Interface Name="Fabrikam.IBar" InterfaceId="12345678-1234-1234-
1234-123456789ABD"/>
      </ProxyStub>
    </Extension>

    <Extension Category="windows.gameExplorer">
      <GameExplorer GameDefinitionContainer="gamecontainer.dll"/>
    </Extension>

    <Extension Category="windows.certificates">
      <Certificates>
        <Certificate StoreName="Root"
Content="Certificates\Root\myroot1.cer"/>
        <Certificate StoreName="Root" Content="myroot2.cer"/>
        <Certificate StoreName="TrustedPeople" Content="mypeer1.sst"/>
        <Certificate StoreName="Issuer" Content="myissuer.cer"/>
        <TrustFlags ExclusiveTrust="false"/>
        <SelectionCriteria HardwareOnly="true"/>
      </Certificates>
    </Extension>

    <Extension Category="windows.digitalRightsManagement">
      <DigitalRightsManagement ConfigFile="PlayReadySetup.xml"/>
    </Extension>

  </Extensions>
```

Application-specific extensions are specified by the **Extensions** element, a child of the **Application** element.

3.2.2 REFERENCES

1. Schema
2. Sample Manifest
3. Refer to the Package Manifest Functional Spec for an overview of all elements in the manifest.
4. Dev Design

3.3 PACKAGE IDENTIFICATION

Package Identification

Identifier	Description	Useage	Example
Package Identity or packageID	Each package is defined by a globally unique identifier known as ***packageID.*** A	Developer friendly representation of package identity	<Identity Name= "microsoft.devx.appx.connect4" ProcessorArchitecture="neutral"

	developer defines this ID in the manifest. It consists of 5 tuples,
	- Name
	- Publisher
	- ProcessorArchitecture
	- Version
	- ResourceId
		that is used to construct the Package Monikers.	Publisher="CN=Microsoft Corporate Root Authority,OU=ITG,O=Microsoft,L=Redmond,S=WA,C=US"
			
			Version="1.0.0.0" />

Package Full Name	The Package Full Name is normalized string form of the packageId. It is constructed in the following way: • The prefixes are removed for each field are removed, e.g. n=, p=, etc. • The publisher information is replaced with a Base32 encoding of the SHA256 hash of the publisher string • An underscore (_) is used to separate the constituent fields PackageFullName = <Name>_<Version>_<Architecture>_< Resources>_<Base32(SHA256(Publisher))> This form makes it suitable for naming objects, such as files, & directories.	The *PackageFullName* should be used to uniquely reference a package on the system, such as when creating the root installation directory for a package.	microsoft.sdksamples.newsreader_1.0.0.0_neutral_en-us_tf9ntdy1vxe7p
Package Family Name	The Package Family Name (PFN) is normalized string form of the packageId. It is constructed in the following way: • Version, Architecture and	The *PackageFamilyName* should be used to reference a package family. It is used when a version-	microsoft.sdksamples.newsreader_tf9ntdy1vxe7p

	ResourceId are removed. • The publisher information is replaced with a Base32 encoding of the SHA256 hash of the publisher string • An underscore (_) is used to separate the constituent fields PackageFamilyName = <Name>_<Base32(SHA256(Publisher))>	independent name of a package needs to be referenced, such as when creating the root state directory for a package.	
Package Relative AppId (PRAID)	The "ID" attribute on the Application element within the manifest is known as PRAID. This string uniquely identifies an application within a package (if a package contains more than one app). This is a required attribute.	It is used to construct the AppUserModelID	Connect4.Web
AppUser ModelID	The AppUserModelID (AUMID) uniquely identifies an application on the system. It is constructed in the following way: AppUserModelID = <PFN>_<PRAID>	Its purpose is to identify the app uniquely to send notification, toasts, etc. to the specific app.	microsoft.sdksamples.newsreader_tf9ntdy1vxe7p_Connect4.Web

3.4 APPLICATION PACKAGES, FRAMEWORK PACKAGES & DEPENDENCIES

A package is the smallest unit of deployment, management, and servicing of software in the Windows 8 application model. A package is essentially an OPC-based .ZIP container that includes a package manifest and a set of binaries. There are two types of packages: application packages and framework packages.

An application package can contain one or more applications and their associated binaries and all supported language resources. A package can contain a mix of app types, such as HTML5 (WWA), WinRT UI (aka Jupiter), and DirectX. Although each application is referenced in the package manifest separately (and must have its own tile that appears separately), management operations are performed on the package as a whole and not individual applications it contains. For example, if an installed package contains two apps, solitaire and backgammon, a user will be unable to uninstall one without also uninstalling the other. All apps in a package run in the same Application Container and thus are able to communicate with each other directly. Apps in different packages must rely on a restricted set of OS-provided communication mechanisms, a byproduct of the apps being in separate Application Containers. See the ***Error! Reference source not found.*** section of this document for more information.

Framework packages contain code that's listed as a dependency by one or more application packages. Framework packages are identified by the Framework element in the package manifest.

```
<!--PROPERTIES-->
<Properties>
  <Framework>true</Framework>
```

By design, frameworks do not contain application tiles. The App Store will support a highly controlled, small list of frameworks for Windows 8 that are all 1st party (e.g. Microsoft produced) – Corsica (JavaScript Framework), C Runtime, and DRM are examples.

The follow list describes some of the other rules regarding dependencies that Modern Applications must follow:

- Application packages can declare dependencies on framework packages so that their installation is required for the application to run. The installation of a framework package happens seamlessly as part of application acquisition from the App Store.
- Framework packages cannot declare dependencies.
- Multiple architecture-specific framework packages should use different package names to avoid conflicts between applications.
- Only one version of a particular framework will be on a user's machine at any given point; all apps that list a dependency on that framework use this single instance. In other words, if Corsica v9 is an "update" to Corsica v8, it's the responsibility of the framework developer (in this case, the Corsica team) to ensure Corsica v9 doesn't break compatibility. So apps A and B that both have dependencies on Corsica v8 will both automatically start using Corsica v9 as soon as an app (app C) that requires it is installed—assuming that Corsica v9 is an update to Corsica v8. If Corsica v9 includes breaking changes, it's the framework developer's responsibility to publish the new version as a separate, new framework package. In that event, apps A and B would stay dependent on Corsica v8 while app C would depend on Corsica v9; Corsica v8 and Corsica v9

would be two entirely separate framework packages that would need to be serviced and maintained independently by the framework developer.

- Packages declare the OS requirements (such as Windows version, service pack level, and architecture) in the manifest as prerequisites.

3.5 CONFIDENCE

Users care about the reliability of their systems, the security of their data, and the privacy of their information. They expect Microsoft and Windows to protect them. Our users have learned over time that applications that are installed and run on the local PC are potential threats, whether they are directly malicious, or simply subject to attack by malicious code. This has caused users to lose confidence in traditional Windows applications and turn to online applications instead. They believe that an application hosted in a browser frame is safer than an application that is installed locally.

The Windows 7 application trust model grants applications full access to the user's private data. This model is referred to as "full trust." This model worked reasonably well when PCs were mostly disconnected, and software acquisition channels were limited. Throughout the internet revolution, Windows users have learned the hard way that installing software on their PCs is not safe. So, they are reluctant to install desktop applications on their PC unless they believe the application is safe. This safety is inferred primarily through brand recognition, which reduces the ability for smaller developers to make money on the Windows platform, because they are more likely to be "unknown" and thus "untrustworthy."

The Windows 8 confidence model increases user confidence and control through the following features:

1. The Application Container
2. A rigorous testing process for the App Store
3. Reputation services to help the community weed out poor applications
4. An update service that delivers updates for store apps
5. As a last resort, revocation: there is a "kill-bit" mechanism to handle bad apps

3.5.1 APPLICATION CONTAINERS

An Application Container is a new security context within which Modern Applications execute. While not a "security boundary," application containers help protect users from attack by running with greatly restricted capabilities on Windows 8, a set of capabilities subordinate to the user's permissions. And while application containers do not protect against all forms of attack, they do significantly reduce the ability of an attack to install malicious code or compromise the user's data and privacy. .

The Application Container design deliberately constrains modern applications from accessing resources that are deemed important for preserving the user's privacy and data. The design depends on setting and enforcing these restrictions through ACLs and brokers. The intent is to give developers a rich enough environment to be able to develop compelling applications while also increasing the confidence customers have in downloading and using these applications.

Application Containers are aligned for the Windows 8 application model's unit of deployment and servicing: a package. Thus, applications within a package run in the same Application Container while applications from separate packages run in separate Application Containers.

Within an Application Container, applications are granted certain rights. For example, they get access to the CPU to run code. They get access to a set of "safe" devices, such as input methods like the keyboard and touch interaction. They can communicate to each other using OS-supported IPC mechanisms, such as RPC and named pipes, and, they get access to a private storage location ACL'd for access only by that Application Container. Applications can use the **GetTempPath** function to retrieve a location for storing temporary files.

```
DWORD WINAPI GetTempPath(
  __in   DWORD nBufferLength,
  __out  LPTSTR lpBuffer
);
```

For more details about the GetTempPath function, see http://msdn.microsoft.com/en-us/library/aa364992(VS.85).aspx

3.5.2 CAPABILITIES

By default, there are a set of capabilities not allowed for apps running in an Application Container that modern applications likely need in order to deliver compelling and useful end-user experiences. Examples include access to libraries (pictures, music), access to networks (Internet, intranet), and access to devices (webcam, location). Developers declare these needed additional capabilities in the app's application manifest. The App Store processes the declarations during the onboarding and publishing process, validating that they're correct and surfacing a small set of relevant information to the user so that the user can make an informed decision about the app at acquisition time. When the user acquires an app, they implicitly consent to the app being able to use the declared set of capabilities. The Application Container also enforces the capabilities declarations specified in the application manifest during runtime. An application running in an Application Container can only ever get access to capabilities declared in the manifest; attempted access to non-declared capabilities will result in access denied error messages. If an application has a need to add additional capabilities after publishing and deployment, the app developer must author a new application manifest with the additional capabilities and re-publish the application to the App Store. Additionally, users will need to "approve" the acquisition of that updated version, as the capabilities of the application will be changed.

The follow is an example of capability declarations in a manifest.

```
<!--CAPABILITIES-->
  <Capabilities>
    <Capability Name="picturesLibrary"/>
    <Capability Name="videosLibrary"/>
    <Capability Name="musicLibrary"/>
    <DeviceCapability Name="webcam"/>
    <DeviceCapability Name="A3A8783C-4844-4b09-9E7A-9E8003D4E20B"/>
  </Capabilities>
```

While most capabilities receive user consent at app acquisition time, some capabilities require consent at runtime. For example, sometimes it's OK for an application like Skype to access a user's webcam. Other times, it's not. Having a runtime mechanism for informing the user about webcam access puts the user in control of their application and Windows experience.

The following list describes the different capabilities an application may declare.

Capabilities

Capability	Manifest Entry	Description
Internet Networking - Outbound only	iternetClient	Gives app outbound access to the Internet and the networks in Public places like airports and coffee shops. This is what almost all internet applications need.
Internet Networking	internetClientServer	Gives app inbound and outbound access to the Internet and the networks in Public places like airports and coffee shops. Inbound access to critical ports is always blocked. capabilityInternetClientServer is a superset of capabilityInternetClient. Hence you do not need to declare both.
Home/work Networking	privateNetworkClientServer	Gives app inbound and outbound access to the networks of user's trusted places like home and the enterprise he works for. Inbound access to critical ports is always blocked.
Pictures Library Access	picturesLibrary	Gives app read/write access to user's pictures library through specific runtime APIs.
Video Library Access	videosLibrary	Gives app read/write access to user's video library through specific runtime APIs.
Music Library Access	musicLibrary	Gives app read/write access to user's music library through specific runtime APIs.
Document Library Access	documentsLibrary	This capability provides programmatic access to files in the users Documents library, filtered to the file types declared by the application. For example, Word can enumerate and open .docx files, but would not be able to programmatically access .xlsx files. The ACLs on the document library will not be changed; an application must use the file broker APIs. Using the file picker still provides access to any file, regardless of file type.

Removable Storage	removableStorage	This capability provides programmatic access to files on removable storage (USB keys, external hard drives, etc.), filtered to the file types declared by the application. For example, Word can enumerate and open .docx files, but would not be able to programmatically access .xlsx files. The ACLs on the devices will not be changed; an application must use the file broker APIs. Using the file picker still provides access to any file, regardless of file type.
Default Windows Credentials	defaultWindowsCredentials	A capability that indicates the application can use a user's domain credentials to access intranet resources. This capability must be enabled through a group policy before it can be used.
Shared User Certificates	sharedUserCertificates	A capability that allows an application to access software and hardware certificates (i.e. smart card certs). When this capability is invoked at runtime, the user must take action (insert card, select certificate, etc.)
Immersive Applications	SECURITY_CAPABILITY_RUN_IMMERSIVE	A capability that indicates the application is an immersive application, and bound by z-banding restrictions. This capability is never presented to the user and is automatically applied for all AppX packages.

3.5.3 BROKERS

Applications can access capabilities that exist outside of the bounds of Application Containers by using brokers. Brokers are simply pieces of code running with the full permissions of the user that are ACL'd so that they can be called from Application Container code.

Guidance on how platform API developers can author brokers can be found at the following location:

http://windows/windows8/DevX/AppX/Shared%20Documents/Trust/APITrustGuidance.docx

A commonly asked question regarding Application Containers is whether it's possible to run apps in a more restrictive or less restrictive sub-container. Unfortunately, no concept of more restrictive sub-containers exists. In other words, it's not possible to establish parent-child relationships between Application Containers in order to run less-trusted external code or data in a more isolated execution environment.

On the other end of the spectrum, applications that require fewer restrictions than Application Containers are discouraged from creating broker components that circumvent the Application Container, as doing so limits the ability of the Windows 8 app model to provide a secure experience. For example, the Application Model requires that a developer declare access to the web cam in order to make use of it; this capability is disclosed to the user at acquisition, and can be further controlled through privacy settings. A 3rd party could decide this restriction is too onerous and attempt to install a broker that grants access to the webcam stream without providing disclosure or control to the end user. The end-to-end Application Model and tooling story imposes the following restrictions to constrain such implementations:

- The Windows Runtime tools will only support 3rd party creation of Base Trust components that run in an Application Container and cannot be used to deliver brokered capabilities.
- The application deployment model will not support installation of desktop (full-trust) brokers.
- The application deployment model will not enable dependencies on 3rd party full-trust brokers.
- The App Store will not accept or distribute applications that have dependencies on 3rd party full-trust brokers.
- The Windows SDK will not provide sample code or instructions for building brokers for modern applications. All brokers available to Windows 8 applications will be first party brokers, included in-box.

3.5.4 REFERENCES

1. http://lowbox

3.6 WWA HOST

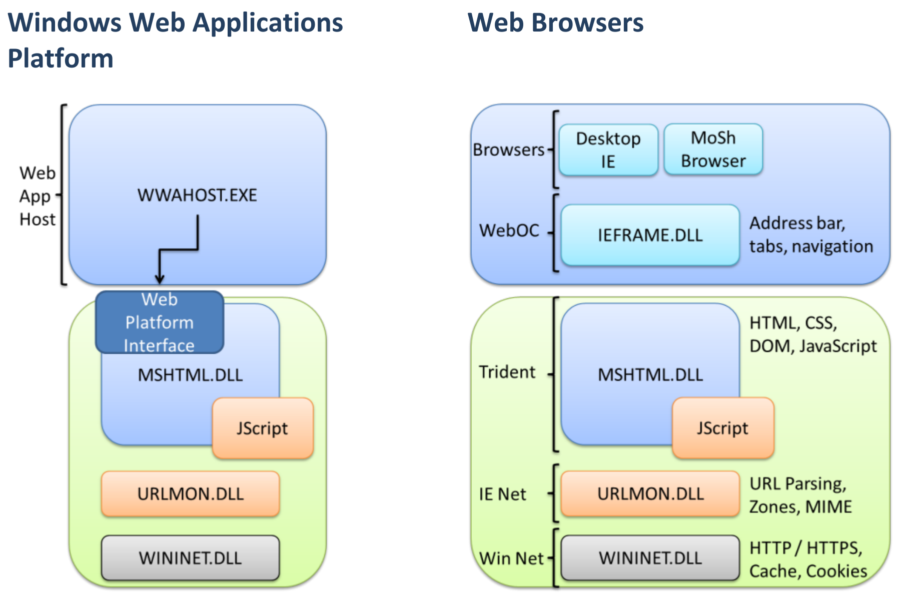
All Windows Web Applications (WWAs) run in the WWA Host. A WWA is a Windows application written in HTML, CSS, and JavaScript that is either completely contained in a package or partially contained in a package with some content being delivered from the web. The WWA Host is a specialized component of the MoSh that does the following:

- Hosts and configures the Web Platform HTML5 rendering component
- Manages when the Web Platform renders to screen and provides the user experience for error conditions
- Provides Windows Runtime (WinRT) integration
- Expands and constrains the capabilities of JavaScript
- Bridges certain MoSh contracts between the Windows and JavaScript environments
- Manages navigation events
- Provides telemetry about WWA usage and crash data

3.6.1 WEB PLATFORM

The WWA Host is not a browser, although it uses the core rendering engine and capabilities of IE. The WWA Host has the ability to navigate to web content and render HTML, CSS, JavaScript, and SVG, but it doesn't have traditional web browser features, such as tabs, an address bar, or some security and reliability features such as the smart screen filter. The WWA Host uses a private, Windows-internal interface from IE to obtain capabilities suited to the needs of a web application platform, called the Web

Platform Control. This interface differs from the hosting interfaces the IE and MoSh browsers themselves use uses and other traditional interfaces of Internet Explorer, such as WebOC and Trident.



The WWA Host hard-codes many settings of the Web Platform to insure uniform behavior across applications and to simplify the developer experience so that developers can rely on a consistent set of behaviors and not have to code for multiple configurations.

3.6.2 CONFIGURATION

The WWA Host configures a number of settings that control the basic behavior of the Web Platform, and there is no way for an application developer or system administrator to override these settings (even through Group Policy). There are other settings that the WWA Host obtains from IE; these settings can be overridden by Group Policy or user changes to settings within IE. For example, if a user changes their proxy in IE, it will also affect all WWAs, but changing a Zone security setting has no effect on the WWA Host.

The WWA Host configures many settings; the following are some of the most important:

1. **Security Settings** – These are the settings you see under Zones in IE.
2. **Cookie Settings** – The user cannot turn off cookies, although all cookies are isolated on a per Application Container basis. To clear cookies, the user uninstalls the WWA.
3. **Standards** – Many non-standard technologies, such as DHTML, doc hosting (how Adobe .PDF documents get displayed "inside" IE rather than launching a separate Adobe app), and IE binary

behaviors are disabled. The WWA Host displays content using the latest web standards support available in the platform, the IE10 Standards Document mode.

3.6.3 HTML, CSS, & JAVASCRIPT

WWA code is written primarily in HTML, CSS, and JavaScript, and can optionally include third-party WinRT objects (see below). There are a number of resources for building these applications:

- Building High-Performance WWAs – DevX Wiki (forthcoming)

TBD: CSS Transforms & Animation Info from PAC

3.6.4 WWA CHARACTERISTICS

There are a number of behavioral differences between authoring a WWA and authoring a typical website. However, most capabilities and coding practices carry over directly from Internet Explorer. The following table lists some of the features supported by WWAs and some of the browser features that are not supported.

Supported Features	Unsupported Browser Features
HTML, CSS, JavaScript, SVG	Address Bar
WinRT access (local compartment only)	Tabs and Tabbed Browsing
Unrestricted XHR (local compartment only)	Toolbars, Explorer Bars, and Browser Helper Objects (BHO)
Application Container	Accelerators and WebSlices
Modern app styling (mostly provided by the default style sheet used by WWAs)	Status Bar
	Favorites
	Navigation stack (History and Breadcrumbs)
	Restore Last Session
	Process Isolation (LCIE)
	IE Crash and Hang Recovery
	Pop-up Blocker (no pop-ups are allowed in WWAs)
	InPrivate Browsing
	VBScript
	Doc Hosting
	Legacy ActiveX

3.6.5 LOCAL CONTEXT & WEB CONTEXT

A web application acquired from the Windows store contains, at a minimum, a manifest that describes the structure of the application and how the user starts it. At runtime, the web code of the application is hosted by the WWAHost. The application code itself may be a mix of HTML, CSS, and JavaScript files contained in the package, on remote web servers (as is common practice when building mash-ups), or a combination of the two.

Web applications can greatly extend their functionality by accessing Windows Runtime (WinRT) APIs. With great power, though, comes great responsibility. The assets behind the WinRT APIs are something the platform needs to protect. While the Windows 8 confidence model assumes that applications from the Windows store are not actively malicious, the web programming model often relies on dynamic code from web servers, which are wholly separate from the Windows store and its policies. Windows cannot rely on the security of servers hosting JavaScript code that a Windows 8 application depends on, or the connection to these servers. (See *Twitter Fixes Cross-Site Scripting Flaw* for a story about a similar security issue.)

The Windows Web App host restricts the available API surface based on code origin while at the same time making it natural and familiar to web developers to know the context they're operating in and do the right thing.

In essence, there are two execution contexts: code in the **local context** has trusted access to the Windows Runtime while code in the **web context** is untrusted, meaning it runs with restricted capabilities similar to how browsers run web content today (there is no WinRT access, for example).

These restrictions are applied at the document level. One HTML document running as the top-level document may run in a different context than another HTML document running in an IFrame. Such contexts can only interact as two DOMs of different origins—communication is generally blocked except through the HTML5 postMessage function.

```
/*If Document A contains a reference to the contentWindow property of
Document B, script in Document A can send a message to Document B by
calling the postMessage method as follows: */

var o = document.getElementsByTagName('IFrame')[0];
o.contentWindow.postMessage('Hello World');

/*The script in Document B can respond to the message by registering the
onmessage event handler for incoming messages. */

window.attachEvent('onmessage',function(e) {
    if (e.domain == 'example.com') {
        if (e.data == 'Hello World') {
            e.source.postMessage('Hello');
        } else {
            alert(e.data);
        }
    }
});
```

An alternative to using postMessage is to use the eval function from within the local context, likely after retrieving content via XHR. The eval function will evaluate all of the script that is returned from the XHR request or provided by the developer without restriction.

```
…

{ if (xmlHttp.readyState == 4 )
{ element.innerHTML = xmlHttp.responseText;
var x = element.getElementsByTagName("script");

for(var i=0;i<x.length;i++)
{ eval(x.text);
} } }
…
```

3.6.5.1 LOCAL EXECUTION CONTEXT

Execution context is determined by where and how the DOM is loaded. Since the primary (or "top-level") DOM is always loaded from files delivered by application package itself rather than from the web, the default is for this DOM to be loaded in a local execution context. This local execution context has the following characteristics:

- It has full access to the Windows Runtime.
- It can load media resources (audio, video, images) either from the local package or by using http:// or https:// to access remote resources.
- It can retrieve content using XmlHttpRequests (XHR) from either the local package or remote sites without respect to same origin policy.

To help the developer ensure that untrusted content, including remote scripts, are not able to disrupt the proper function of the application, the local execution context ensures compliance with the following restrictions:

- The javascript: protocol is blocked to prevent potentially unsafe remote code from executing.
- HTML documents must be encoded in the UTF-8 codepage.
- Any manipulation of the DOM that can accidentally insert script, most commonly through a method like innerHTML, is checked for potentially unsafe attributes or scripts and these are blocked by default. This particular requirement can be overridden by the developer.

In this execution context, accessing content from the local package normally only uses the ms-wwa:// URI scheme. For example,

```
<script src="ms-wwa:///default.js"/> or <script src="default.js"/>
```

Notice the extra slash after ms-wwa:///. This is an alias for the host name of your WWA, so you don't have to provide the package name as part of the ms-wwa URI scheme. This is consistent with the File URI scheme.

3.6.5.1.1 OVERLOADING LOCAL EXECUTION CONTEXT RESTRICTIONS ON INNERHTML

In some cases a developer may need to override the restrictions on innerHTML (or related functions).

To avoid this exception, there are a few options –

1. Sanitize the content with toStaticHTML() prior to insertion.
2. Create the content explicitly, using createElement() and setAttribute().
3. Use msWWA.execUnsafeLocalFunction() to insert unsafe content.

The 3rd option allows you to pass a function that will be executed without the SafeHTML validation. This should not be used lightly, since it can allow unsafe content into your WWA. If your WWA is manipulating web content that is not fully under your control such as advertisements, blog comments, etc., you should not use this function since it could allow an attacker to introduce script or other unsafe content.

3.6.5.2 WEB EXECUTION CONTEXT

While the default DOM is loaded from the local package, and displayed in the local execution context by default, there are times when an application will want and need to display content from the web free from these restrictions. In most cases, this is done with an IFRAME that has a source of http:// or https://. When the host sees an IFRAME creating using http, it will be displayed in a web execution context. The web execution context has the following characteristics:

- It has no access to the Windows Runtime.
- As with any IFRAME it has no access to the parent DOM.
- Any XHR operations are subject to same origin policy.

Functionally any IFRAME running in a web execution context has similar characteristics as an IFRAME running in a browser. This allows the developer of a WWA to display advertising or other mashup content from remote sites without exposing the application to potentially unsafe content.

3.6.5.3 SPECIAL CASE FOR EXECUTING PACKAGED CONTENT IN A WEB CONTEXT

Normally the web execution context is used for remote content accessed via the http protocol, but there may be times when it is appropriate to load content from your local package but force it to be loaded in a web execution context. This is accomplished by creating an IFRAME that points at source from your local package by using the ms-wwa-web:// URI scheme.

3.6.5.4 NESTED EXECUTION CONTEXTS

Since IFRAMEs can be nested, it is important to understand the rules for nesting IFRAMEs relative to their execution context. As noted above, the top level DOM is always loaded in a local execution context. The following rules apply:

- The top level DOM can include IFRAMEs of any type.
- An IFRAME running in a local context can create a child IFRAME of any type.
- An IFRAME running in a web context can only create other IFRAMEs that will run in a web execution context. Specifically:
    - IFRAMES with a source URI of http://, https://, or ms-wwa-web:// are allowed
    - IFRAMES with a source URI of ms-wwa:// are not allowed.

These rules ensure that an IFRAME running in a web execution context cannot gain access to the Windows runtime simply by creating a child IFRAME that has such permission.

3.6.5.5 COMMUNICATION BETWEEN IFRAMES

All IFRAMEs, regardless of execution context, have limited ability to communicate with or manipulate any DOM they do not own. WWAs support HTML5 web messaging using postMessage() and addEventListener(). An application developer could use this mechanism to signal a request from one IFRAME to execute code in another IFRAME deliberately. This would allow a developer to call a Windows Runtime API in response to an event that occurred in an IFRAME running with a web execution context which could not otherwise access the API. The deliberate nature of this action ensures that untrusted code cannot directly access the Windows Runtime.

3.6.6 WEB NAVIGATION MODEL

WWAs offer a navigation experience that can take the user to pages packaged with the application, to external sites that are defined as being part of the application, and to third party apps or sites that are not part of the application. These navigation actions may be initiated by the user or programmatically in the context of using the application. WWAs support the following URL schemes: “http,” “https,” “ms-wwa,” “ms-wwa-web,” “data,” “mailto,” and “javascript.”

The “ms-wwa” scheme replaces “file” as the mechanism for retrieving resources in locally packaged content. The “ms-wwa-web” scheme also retrieves locally packaged content and runs the specified in the web compartment.

As the developer, you are responsible for defining the navigation experiences within the application, such as the navigation structure and navigation controls. You are also responsible for defining the boundaries of the application; you do so by configuring the **ApplicationContentUris** element in the application manifest.

3.6.6.1 DEFINING THE NAVIGATION DOMAIN

The **ApplicationContentUris** element contains one or more rule entries, each of which contains a match attribute that specifies a URI match and a type attribute that specifies whether the matched URI should be included or excluded from the application’s content.

There are several ways to specify a URI match:

- You can specify an exact hostname.
- You can specify a hostname for which a URI with any subdomain of that hostname is included or excluded.
- You can specify an exact URI.
- You can specify an exact URI that can contain a query property.
- You can specify a partial path and use a wildcard to indicate a particular file extension.

The following example shows application content URIs with three entries.

```
<ApplicationContentUris>
    <Rule Type=”include” Match=”http://www.example.com/”/>
    <Rule type=”exclude” Match=”http://www.example.com/downloads/”/>
    <Rule Type=”exclude” Match=”*.pdf”/>
</ApplicationContentUris>
```

The first rule includes all of http://www.example.com; the next rule excludes anything under the downloads subdirectory of that site; and the third rule excludes any URI that that ends with “.pdf”.

To determine if a URI is in the application context URIs, the WWA host will take the last matching entry and the URI will be included or excluded based on that entry. For example ‘http://www.example.com/file.pdf’ will be excluded based on the above navigation domain since although it matches the first include entry it also matches the last exclude entry. If the first and last entries in the **ApplicationContentUris** were swapped, then the file.pdf URI would match the include rule last and so would be included in the application content. If a URI does not match any entry, it is not in the application content. ApplicationContentUris may contain a maximum of 100 Rule elements.

Note that the ApplicationContentUris also allow any defined URIs to access the geolocation, media capture, clipboard, and download of unknown files that are normally present in the browser. The geolocation and media capture APIs require a corresponding capability in the package manifest, and still require user consent before they can be invoked. Downloading an unknown file (i.e. a .pdf file that cannot be handled by the application) will redirect the user to the browser.

3.6.6.2 THE APPLICATION CONTENT URIS AND OTHER URI SCHEMES

The ApplicationContentUris only applies to http and https schemes. You can use the new packaged content URL scheme (ms-wwa) to refer to any file that is included in the application's package—it's automatically considered a part of the application content.

The JavaScript URI scheme is automatically included in the application's content. The JavaScript URI scheme is of the form javascript:*code*; for example:

```
javascript:alert('hi')
```

When a JavaScript URI is navigated to, the contained script in executed the context of the document that contained the link. Note that that javascript URIs can only be invoked for web content. Locally packaged content may not use javascript URIs due to common security vulnerabilities associated with the URI scheme.

All other URL schemes are outside the application content URIs.

URLs are interpreted in the same manner that IE interprets URLs. This is mostly based on the definition of URI from RFC 3986 but with the addition of support for non-US-ASCII characters everywhere except the scheme name, and IDN in the hostname. Non-US-ASCII characters are allowed in URLs and will not be removed, encoded, or changed in any manner, except for IDN. In the hostname of http and https URLs, non-US-ASCII characters will be normalized according to IDN and must be valid in IDN.

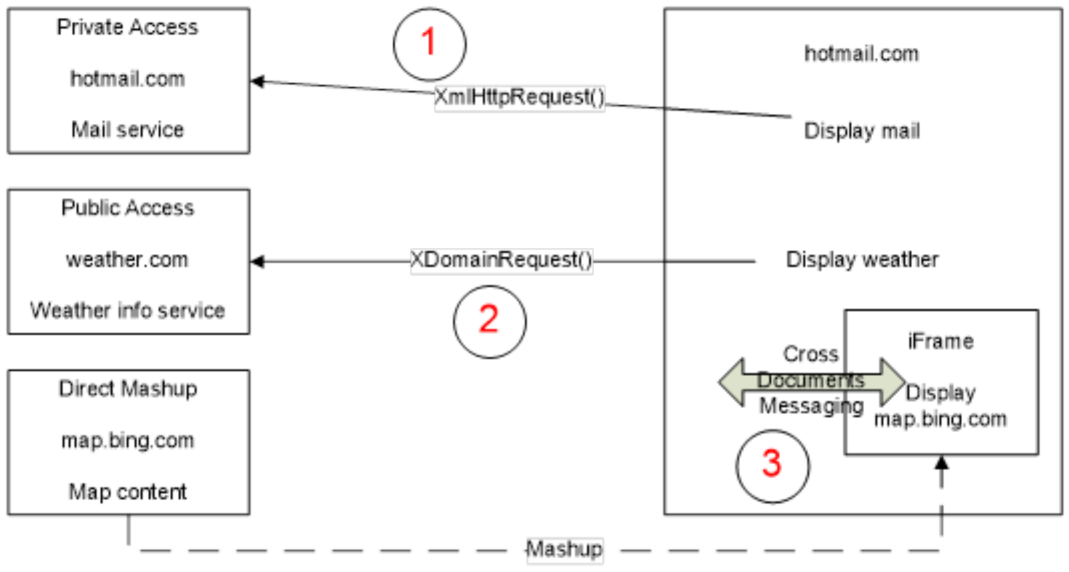
3.6.7 WEB SECURITY, SAME SITE ORIGIN, AND XMLHTTPREQUEST

In the web world, the site of origin (or the origin of a web page) is used to determine the identification of the web page. The concept is used to protect servers from unauthorized access and to create documents that access across server boundaries, such as IFrame documents.

We can consider usage of origin in roughly three groups

- Private resources accessed by same-origin web page. Example (1): Outlook web access uses XHR to bring down a user's inbox / calendar from an OWA server.
- Public resources accessed by different origin web page. Example (2): Service.weather.com has a public method that allows any 3rd party web page to get the current weather data.

- Secure communication between host document and document IFrame. Example (3): Map.bing.com has "map" mashup contents.
    - Variation: Secure communication between documents in different IFrames.



The WWA platform enables extended communication mechanisms, particularly in the local compartment. The following table describes these mechanisms.

Mashup Method	Works with different site of origin?	Example	Comment
XmlHttpRequest	Yes in local compartment, No in web compartment		From the local compartment, XHR is allowed to anywhere. From HTML documents with a web context, the same origin policy restriction applies.
JSON via <script> tag	Yes	Flickr JSON API, Twitter API	The downside of this is approach is that it lets the script tag's target site run whatever script they like on the mashup site.
IFrame	Yes	Facebook Live Conversation	This is a limited form of mashup since there is usually only communication from the hosting page to the framed page when creating the frame but no additional communication.
IFrame + cross	Yes		Same as the preceding approach,

document messaging			but the script running in the hosted page and framed page may communicate if each allows communication with the other's site of origin. Cross document messaging is available in the WWA platform.
IFrame + URL fragment hack	Yes	Facebook Platform JS API	IFrames communicate between the host and the framed page via the one thing that's visible to the host page and controllable by the framed page: the fragment of the framed page's URL. This is a hack and doesn't have the control that cross doc messaging provides, but still works in WWAs. The postMessage method is the preferable approach, though.
Server side code or native client side code (usually RESTful XML web API with no cross origin resource sharing)	Yes	Netflix REST API, Flickr REST API, Facebook User Data REST API	WinRT networking APIs are available to facilitate this.

3.6.8 ERROR HANDLING AND DISPLAY

When a navigation error occurs, the WWA Host will display an error page. The host will provide a generic error page for top level navigation errors, but applications can choose to provide their own. It is recommended that applications provide a custom html page that can handle errors, in order to give users a more fluid experience.

The WWA Host will redirect to this page whenever a navigation error occurs, precipitated by one of these conditions:

- **Receipt of an error HTTP status code**—the most common being 404 (not found) or 500 (internal server error)
- **Receipt of an error HRESULT when talking to networking layer** (WinInet)

  - Web compartment attempts to navigate to a disallowed page in local package—the web compartment can only navigate to a local page it has previously visited in this session; this becomes a standard 404 HTTP status code.
  - Web compartment attempts to navigate without Internet capability—navigation must conform to Application Container restrictions; this becomes a standard 404 HTTP status code.
  - Local compartment attempts to navigate to content outside its navigation domains.
  - Use of a non-standard scheme—Only http://, https://, ms-wwa://, and ms-wwa-web:// schemes are allowed

Characteristics of a custom error page provided by the application:

- The page name must be "wwa-error.html"
- The page must be located in the "root" directory of the package

Characteristics of the default error page provided by the host:

- Only loaded when top level navigation error occurs
- For an Iframe error, the host will show a blank page in the absence of "wwa-error.html"
- Has a back and close button.

This error page must determine the reason for the error, and then determine how to handle the error and return to the previous state or display an appropriate error message to the user. Note that navigation errors can occur either in the top-level window or in an IFrame. The error page is only displayed in the frame or window that experienced the error.

The WWA Host makes certain information about the error available to the error page. These are encoded in the manner described in the HTML4 spec for form submissions encoded as application/x-www-form-urlencoded documentation. The encoded values include:

- **proxyUrl:** the URL to resolve the issue for parental controls or a coffee shop or hotel Terms & Conditions Agreement
- **httpStatus:** the HTTP status code (if applicable) as a decimal integer
- **failureName:** the name of the **HRESULT** of the failure exception from WinInet for the INET_E_ prefix (e.g. AUTHENTICATION_REQUIRED); if the HRESULT is not an INET_E_* failure or if it is one of the internal IE errors, such as INET_E_RESERVED*, then the string "UNKNOWN," concatenated with the hex value of the HRESULT, is returned instead (for example, UNKNOWN80040005)
- **failureUrl:** the URL that precipitated the navigation error; the failureUrl appears last on the query string and will be truncated if the error page URL with this information exceeds the maximum length for an URL.

Note: An application may avoid many error conditions by only navigating to content in the local package or by using the HTML5 Application Cache.

The following example shows a simple error page:

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Error</title>
    <link href="css/error.css" rel="stylesheet" type="text/css" />
    <script src="js/jquery-1.4.3.js" type="text/javascript"></script>
    <script src="js/error.js" type="text/javascript"></script>
</head>
<body>
    <h2>Error</h2>
    <div id="content">
        <p>The application failed to navigate.</p>
        <div id="failureName">
            <span>FailureName: </span>
            <span id="failureNameValue"></span>
        </div>
        <div id="httpStatus">
            <span>HttpStatus: </span>
            <span id="httpStatusValue"></span>
        </div>
        <div id="failureUrl">
            <span>FailureUrl: </span>
            <span id="failureUrlValue"></span>
        </div>
        <div id="proxyUrl">
            <span>ProxyUrl: </span>
            <span id="proxyUrlValue"></span>
        </div>
    </div>
</body>
</html>
```

References:

1. Status codes and failure HRESULTs on MSDN
2. Design Spec for WWA Navigation – Specifies how each HTTP status code and HRESULT is handled, in detail
3. Functional Spec for WWA Navigation

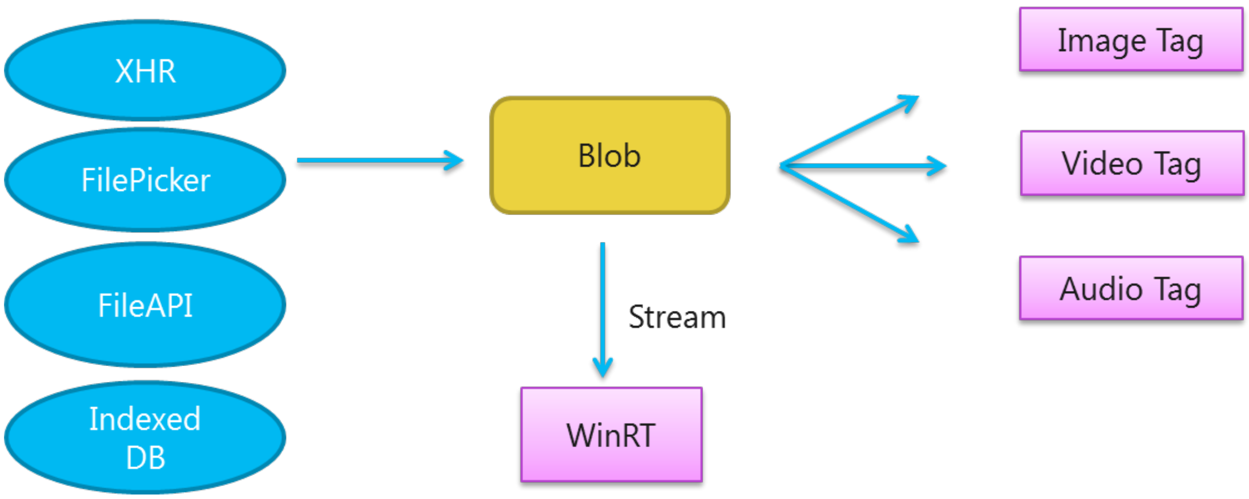
3.6.9 FILE DOWNLOADS

Unlike native languages, JavaScript provides no way to directly interact with and manipulate binary data. Instead of forcing developers to interact with binary data at the byte level, JavaScript developers pass this data to APIs that can manipulate the data for them.

To support binary data manipulation, the web standards defined the blob interface. The blob interface wraps binary data and provides a way to pass the data around. The blob interface was originally defined as part of the File API as a way for applications to use images, videos, and audio that may come from files, but has since been expanded to represent binary data in general. The blob functionality is not currently present in Microsoft technologies (IE, WebOC, or WWAHost), and is essential to enable high-powered web applications.

The Windows Runtime projects many data formats into JavaScript that are not directly consumable by the language. JavaScript developers have no way to use streams, interfaces, or many other data types. The blob interface solves this problem by wrapping the data and providing an API for accessing it. A web developer uses blob as the means to transfer data to other applications and system components, and lets those pieces determine how to use the underlying data represented by the blob. The following list describes some common uses:

- MoSh contracts. Applications want to be able to share and pass data around (playTo, share, and so on) and blob will be a part of this process.
- FileAPI. Reading and writing files will generally require converting to and from blob.
- XHR L2. The XMLHttpRequest Level 2 W3C specification defines an API that provides scripted client functionality for transferring data between a client and a server. The level 2 specification defines support for binary send and receive using blob.



3.6.9.1 LOADING A BLOB

The following example shows how to use XHR to load a blob image into an image tag.

```
var xhrRequest;
function readyStateCallback()
{
   if (xhrRequest.readyState == 4 && xhrRequest.status == 200 )
   {
        var blob = reqXML.response;
        var myBlobUrl = window.URL.createObjectURL(blob);
       document.getElementById("myImageTag").src = myBlobUrl;
   }
}

function DownloadBlob()
{
    xhrRequest = new XMLHttpRequest();
    if (xhrRequest)
    {
        xhrRequest.open("GET", "http://myserver/myimage.jpg", true);
        xhrRequest.responseType = "blob";
        xhrRequest.onreadystatechange = readyStateCallback
        xhrRequest.send(null);
    }
}
```

The next example uses a blob to set an image source.

```
var myImage = document.getElementById("myImageTag");
myImage.src = window.URL.createObjectURL(myBlob);
```

3.6.9.2 RETRIEVING ALL FILES IN THE MUSIC LIBRARY, SORTED BY ARTIST, ALBUM, TRACK, AND WALKING THE LIST

The following example launches a file using a blob.

```
var library = Windows.Storage.KnownFolders.musicLibrary;
library.getFilesAsync(Windows.Storage.CommonFileQuery.orderByMusicInfo).th
en(function(files) {
    files.forEach(function(file) {
           var urlToFile = window.URL.createObjectURL(file);
           //now we can load this URL into an audio tag
    });
});
```

3.6.9.3 DOWNLOADING A FILE USING A BLOB

The following example downloads a file using a blob.

```
var xhrRequest;
function readyStateCallback()
{
   if (xhrRequest.readyState == 4 && xhrRequest.status == 200 )
```

```
    {
        var blob = reqXML.response;
       var stream = blob.msRandomAccessStream;//this is a winRT random
                                        //access stream that can be used to
                                        //write to a StorageItem stream
    }
}

function DownloadBlob()
{
     xhrRequest = new XMLHttpRequest();
     if (xhrRequest)
     {
         xhrRequest.open("GET", "http://myserver/myfile", true);
         xhrRequest.responseType = "blob";
         xhrRequest.onreadystatechange = readyStateCallback
         xhrRequest.send(null);
     }
}
```

3.6.9.4 LAUNCHING A FILE USING A BLOB

The following example launches a file using a blob.

```
function LaunchFileHandler(file)
{
   Windows.UI.Activation.Launcher.LaunchDefaultProgramForFile(file);
}
```

3.6.9.5 ADDITIONAL EXAMPLES

For addition examples, see the Blob DDI Spec.

3.6.9.6 WORKING WITH STREAMS

In addition to Blobs as noted in the previous section, WWAs will also support the notion of MSStream. MSStreams are similar to Blobs, but have a few key differences, notably no known size. The advantage of Streams is that you are able to get them from XHR while the contents are still being downloaded (For a Blob, you must wait till readyState 4, which means the content is fully downloaded). This is the key way to load videos via XHR without forcing the user to wait for the entire video to download.

The stream can be downloaded as follows:

```
var xhrRequest;
function readyStateCallback()
{
   if (xhrRequest.readyState == 3)
   {
        var MSStream = reqXML.response;
        var UrlToStream = URL.createObjectURL(stream);
        //you can now load the url to this stream into a tag.
        //note: This stream was received in readystate 3, unlike the blob
        //which required readystate 4
   }
}

function GetStream()
{
    xhrRequest = new XMLHttpRequest();
    if (xhrRequest)
    {
        xhrRequest.open("GET", "http://myserver/myfile", true);
        xhrRequest.responseType = "ms-stream";
        xhrRequest.onreadystatechange = readyStateCallback
        xhrRequest.send(null);
    }
}
```

Additionally, we provide ways to read the streams data as a Blob. This shows the code necessary to convert an MSStream to a blob by use of the StreamReader. This function assumes that you already have an MSStream object. The StreamReader is an asynchronous interface, and was designed to give developers the same coding pattern as the W3C FileReader.

```
//function which takes an msStream obtained through XHR or window.msObject
function readAsBlob(stream) {

  var reader = new MSStreamReader();

  // Read file into memory
  reader.readAsBlob(stream);

  // Handle progress, success, and errors
  reader.onprogress = updateProgress;
  reader.onload = loaded;
  reader.onerror = errorHandler;
}

function updateProgress(evt) {
  if (evt.lengthComputable) {
    // evt.loaded and evt.total are ProgressEvent properties
    var loaded = (evt.loaded / evt.total);
    if (loaded < 1) {
      // Increase the prog bar length
      // style.width = (loaded * 200) + "px";
    }
  }
}

function loaded(evt) {
  // Obtain the read file data
  var blob = evt.target.result;
}

function errorHandler(evt) {
  if(evt.target.error.code != 0) {
    // The stream could not be read
  }
}
```

3.6.10 JAVASCRIPT CHANGES

The following table details the changes that will be occurring to the browser properties that are accessible by JavaScript code running in the WWAHost:

JavaScript Changes		
navigator	AppCodeName	Return WWAHost/1.0
navigator	appMinorVersion	Return 0
navigator	appName	Return WWAHost/1.0
navigator	appVersion	Return 1
navigator	browserLanguage	Returns the OS language
navigator	userAgent	User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0; WWAHost/1.0;)

Additionally, the following set of JavaScript features as found in the browser's IE10 Standards Mode have been deprecated within WWAs.

- **Conditional compilation.** Conditional compilation is a native solution available to only Internet Explorer which offers web developers a way to find out a number of values specific to the browser environment. Detailed list of such variables and the usage can be found in MSDN. These typically are used to detect if the browser being used is Internet Explorer and handle the behavioral differences of features between browsers. This Microsoft specific extension is not required in the context of WWAs as they will always be running on same web platform.
- **getVarDate().** The getVarDate method is a Microsoft specific extension on the Date object and is used when code interacts with COM objects, ActiveX objects, or other objects that accept and return date values in VT_DATE format. This method is primarily is used to expose a JavaScript date as an OLE date.
- **Enumerator().** Enumerator objects provide an alternative mechanism for iterating over the elements of Array instances and certain host objects that are [Collections]. (e.g. iterating XML DOM). For all JavaScript array or regular objects, For…In enumeration is the standard way to enumerate the properties. Enumerating host/external objects that do not support For..In convention will require the host/external objects to pass serialized JSON strings to JavaScript.

- **VBArray().** VBArrays provide access to Visual Basic safe arrays in JavaScript. VB is not a supported script language within the WWA Host.
- **HTML Wrappers.** The String.prototype functions no longer support APIs like String.prototype.bold() and String.prototype.italics(). All styling should be done via CSS (or HTML).

3.6.11 OFFLINE EXECUTION

Fully packaged WWAs and those that take advantage of HTML5 App Cache are capable of running while not connected to the internet.

3.6.12 STATE & SETTINGS

WWAs provide several choices for where and how to save state and settings. These include the following:

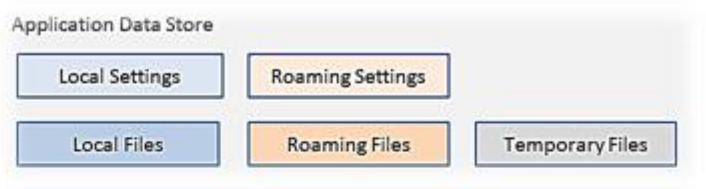
Storage Location	Pros	Cons
State Manager (WinRT)	1. Accessible from both JavaScript and native code, including other WinRT objects 2. Provides structured, unstructured, and temp state locations 3. Can participate in Windows Roaming	1. Not a web standard
Cookies	1. Web standard 2. Travels to web server with each request 3. Isolated from IE and other apps' cookies and only deleted when expired or when the app is uninstalled	1. Limited amount of data storable
Web Storage / DOM Storage	1. Essentially unlimited amount of data storable, limited only by performance considerations since it maps to a single XML file internally 2. Structured data solution	1. Not fully compliant with latest standard; only stores strings (JSON encode/decode required) 2. No unstructured data
File System (WinRT)	1. Full user file system stores available	1. Not a web standard 2. No structured data

	2. Automatic access to application state locations, brokered access to user data 3. Remembers permissions granted by the user	3. No temp file cleanup 4. File parsing is required 5. Binary file writing is limited
Indexed DB	1. Web Standard 2. Structured data solution	1. No unstructured data
Third Party DB (WinRT)	1. Supports full SQL syntax 2. Very familiar (SQL CE, MySQL)	1. Developer must write their own WinRT wrapper 2. No unstructured data

3.6.12.1 STATE MANAGER OVERVIEW

The State Manager is the recommended means of storing state and settings for MoSh applications. MoSh applications can use it to save and restore runtime state, user preferences, settings, performance caches, and other user data. Structured state storage is provided in the form of settings. Unstructured state is provided in the form of files.

A MoSh data store is always provisioned as follows:



These stores are created when the app is installed and deleted when the app is removed. They are accessible via the **Windows.Storage.ApplicationData.current** WinRT class.

3.6.12.2 SETTINGS

The State Manager provides MoSh apps with access to settings in its local and roaming settings containers through two static properties: **Windows.Storage.ApplicationData.current.LocalSettings** and **Windows.Storage.ApplicationData.current.RoamingSettings**. These properties both have a childrens

The following examples show how to use JavaScript to read, delete, and write State Manager settings:

```
// SETTING READ: Get value of "window-background-color" roaming setting
var windowBackgroundColor =
    Windows.Storage.ApplicationData.current.RoamingSettings.values.lookup(
 "window-background-color");

// SETTING DELETE: Delete the value of "window-text-font" roaming setting
```

```
Windows.Storage.ApplicationData.current.RoamingSettings.values.remove(
 "window-text-font");

// SETTING READ: Get the value of the "app-launch-count" local setting
var appLaunchCount =
    Windows.Storage.ApplicationData.LocalSettings.values.lookup(
 "app-launch-count");

// SETTING WRITE: Set the value of the "app-launch-count" local setting
Windows.Storage.ApplicationData.LocalSettings.values.insert(
 "app-launch-count", appLaunchCount++);
```

3.6.12.3 FOLDER & FILES

The State Manger provides MoSh apps with easy access to their local, roaming and temporary data folders through three static properties: **Windows.Storage.ApplicationData.current.LocalFolder**, **Windows.Storage.ApplicationData.current.RoamingFolder,** and **Windows.Storage.ApplicationData.current.TemporaryFolder**. File methods defined by the **Windows.Storage.IStorageItem** may be called directly on any of these folders to access the application's data files.

The following example shows how to retrieve data from the **RoamingFolder**.

```
// FOLDER OPEN: open the roaming folder in the application data store.
var folder = Windows.Storage.ApplicationData.current.RoamingFolder;

// CREATE FILE STREAM: create an sample CSV-format data file in the
temporary folder.
var fileStream = folder.CreateChild("address-
book.xml").GetStream(ReadWrite);
// do something with the XML file stream here…

// CLOSE FILE STREAM:
fileStream.Close();
```

An application data folder is represented by the **Windows.Storage.IStorageFolder** interface, a public MoSh API that can represent a folder or a file. For more information on File Access, please see section 4.15 of this document.

3.6.12.4 CONTAINERS

Within the State Manager APIs, the developer may create containers and nested containers to organize their files and settings.

MoSh app developers may create nested container hierarchies up to 32-levels deep with no restriction on tree breadth. To create a nested container, call the parent container's **CreateContainer** method. To delete a nested container, call the parent's **DeleteContainer** method.

The following example shows how to create and delete containers:

```
// Get the local root settings container.
var localSettingsContainer = appData.localSettings;

// Create a named container under the root.
var nestedContainer = localSettingsContainer.createContainer("Nested");

// Write a value into the new "Nested" container.
nestedContainer.values.insert("pi", 3.1415926536);

// Delete a subcontainer.
localSettingsContainer.deleteContainer("Nested");
```

Because setting containers represent a live view of a volatile set of items, they do not support the deterministic enumeration semantics a developer would expect of an instanced, in-memory collection. However, it's sometimes necessary for a client application to take an inventory of a container's contents.

This is enabled via the **GetItemsView** method. This method returns an immutable, in-memory collection comprising the names and type of the items found in the container at the time of the request. The following example demonstrates this method.

```
// ENUMERATE CONTAINER: Obtain an immutable map view of the items
//              currently in the container.

var iterator = container.values.first;

// Iterate through the results
while (iterator.hasCurrent)
{
    // Do something with the result. E.g., write it to the document.
    document.writeln(iterator.current.key + ": " + iterator.current.value
+"<br />");
    iterator.moveNext();
}
```

3.6.12.5 ATOMS

The State Manager introduces the concept of a *setting atom* – a new type of composite setting designed to allow MoSh application developers to save and restore tightly-coupled data sets, such as multiple properties of a single object, with simple and deterministic concurrent access and roaming. Like setting containers and settings, setting atoms are named, created, and deleted relative to a parent settings container.

Setting atoms are optimized for tightly-coupled settings groups that contain small amounts of data. They are expected to perform poorly if used for large, unbounded data sets. Enumerating the settings in an atom is not supported.

Atoms are created and deleted with the CreateAtom and DeleteAtom methods.

The GetValue, SetValue, and DeleteValue methods defined by Windows.Storage.ApplicationDataAtom are syntactically identical to their ApplicationDataContainer counterparts. However, setting operations in an atom are not committed until an application calls the ApplicationDataAtom.Flush method.

The following example creates, sets, and deletes atoms:

```
// ATOM CREATE: Create "window-position" atom in local settings container.
var atom = localContainer.CreateAtom("window-position",
              Windows.Storage.ApplicationDataCreate_Always);

// ATOM SETTING ACCESS: Write some setting values inside the atom.
atom.setValue("upper", 0);
atom.setValue("left", 0);
atom.setValue("lower", 480);
atom.setValue("right", 640);

// ATOM SETTINGS FLUSH: Commit modifications to the settings group.
atom.flush();

// ATOM DELETE: Delete the atom just to demonstrate how it’s done.
localContainer.deleteAtom("position");
```

3.6.12.6 ADVANCED OPERATIONS

When an application is updated, its state schema may change and require versioning and upgrade support. Additionally, an app developer might want to be notified when roaming of state settings occurs, in order to synchronize with the incoming roaming state and settings data. These topics are described in the references that follow.

3.6.12.7 REFERENCES

1. State Manager WinRT Projection Spec
2. StateManager Functional Spec
3. StateManager Design Spec

3.7 VIEW SIZING AND SCALING

Windows 8 is a reimagining of Windows, from the platform to the user experience. A Windows 8-based PC is new kind of device, one that scales from small screens on slates through to medium laptop screens, to large screens on desktops and all-in-ones. A Windows 8-based PC also scales to different pixel densities. It scales from the pixel density of a slate in the marketplace today through to high-end HD Slates. It also will scale all the way to the super high pixel density slates of the future; devices where the size of a pixel is imperceptible to a human eye.

Support for this variety of screens is a unique offering of Windows 8-based PCs. This variety allows for PC manufactures to differentiate themselves with different and innovative form factor designs. Instead of only offering one type of screen, consumers can choose the screen size and pixel density that best suits their preference and usage.

Diversity of choice for PCs drives the broad reach and of the Windows platform with over 1 billion PCs in the market today. Apps built using HTML5 and JavaScript for Windows have the potential to be run on any Windows 8 PCs. Support for this breadth and diversity of PCs could potentially add complexity and concepts to the developer story but Windows 8 has been designed from the ground up to embrace this diversity while making it simple for app developers to accommodate it.

3.7.1 VIEW SIZING

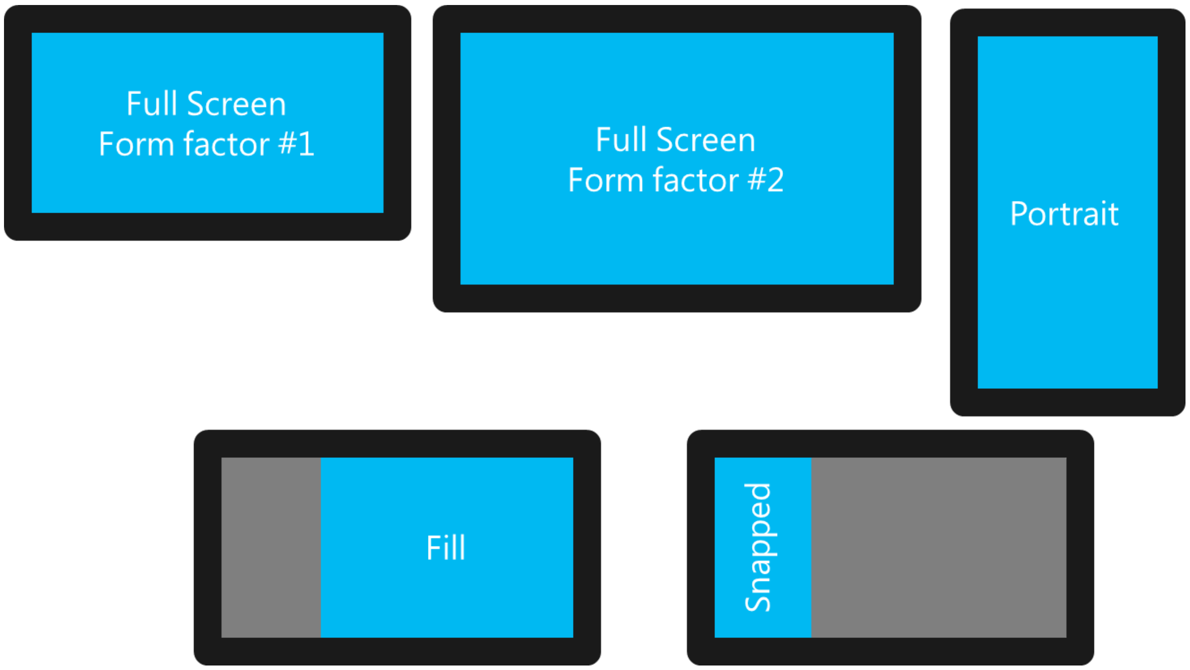
Support for varying screen sizes from different size and shaped slates, laptops and desktop monitors are a unique strength of Windows 8. Besides the different form factors, applications may be resized depending on the orientation of the device or a variety of other factors.

3.7.1.1 WHEN YOUR APPLICATION CAN BE SIZED

Applications are resized in the following circumstances:

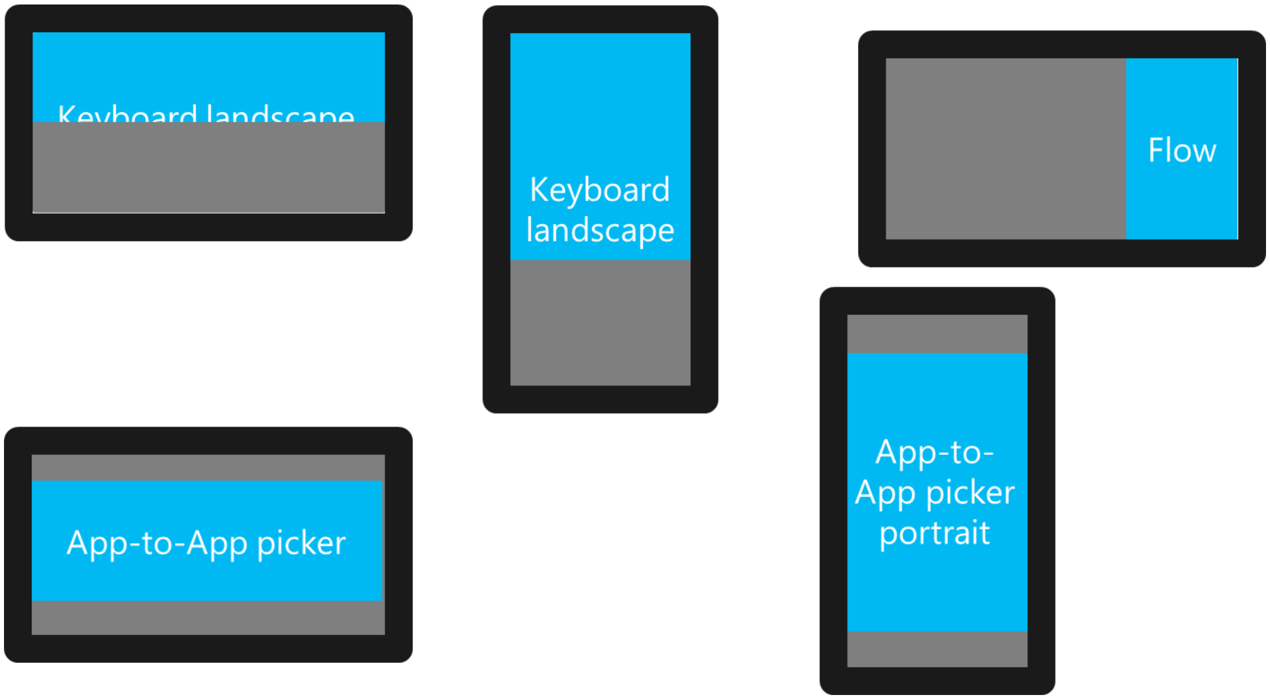
1. The display changes. For example, the display is switched to an external monitor or projector.
2. The computer has multiple monitors and the application is moved from one to the other.
3. The app is snapped
4. Another app is snapped, decreasing or increasing the space available for the application.
5. The display is rotated.

The following illustration shows how the preceding factors can change size of an application:



Applications may also support several specialized layouts:

1. Keyboard shown
2. App-to-app picking
3. Share flyout
4. Connect flyout



3.7.1.2 ADAPTIVE LAYOUTS

One way that Windows 8 helps application developers build for this varying screen sizes is by support in the application platform for standards-based adaptive layouts. The CSS3 Grid Layout, Flexible Box Layout and Multi-column Layout helps developers effectively use screen-real estate across a variety of devices and resolutions. The CSS3 Grid Layout allows a developer to specify the rows and columns of their layout, each of which can be defined as fixed, flexible or size-to-content. Some of the elements can remain fixed, while others can grow or shrink as the size of different monitors. The CSS3 Flexible Box Layout allows a developer to stack their content in one dimension and the control will ensure that whitespace is distributed equally and predictably. Finally, you can use CSS Multi-column layout to arrange content into multiple columns on the page. This approach is used by newspapers and magazines in the print world, which makes text easier to read and track from line to line by organizing the content into multiple parallel columns. All of the templates have been designed to support different sized screens by default by using these layout constructs and leveraging controls that fill the screen like the ListView control.

In an adaptive layout, content and margins reflow to accommodate size changes.

- Provide a full immersive experience.
- Have the potential to scale across all devices and device families.
- Take advantage of screen real estate.
- Can add more complexity to application; all of today's desktop apps are fluid.

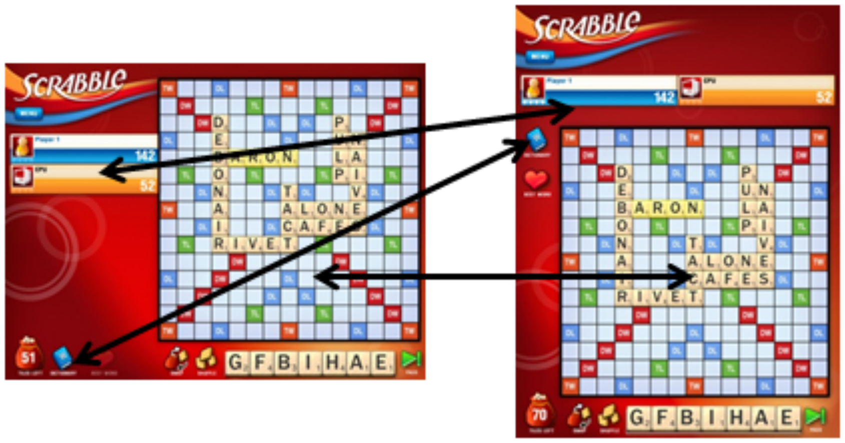
3.7.1.3 ADAPTIVE LAYOUT COOKBOOK

There are three considerations to think about when building applications that are adaptive and maximize the all of the available screen real estate.

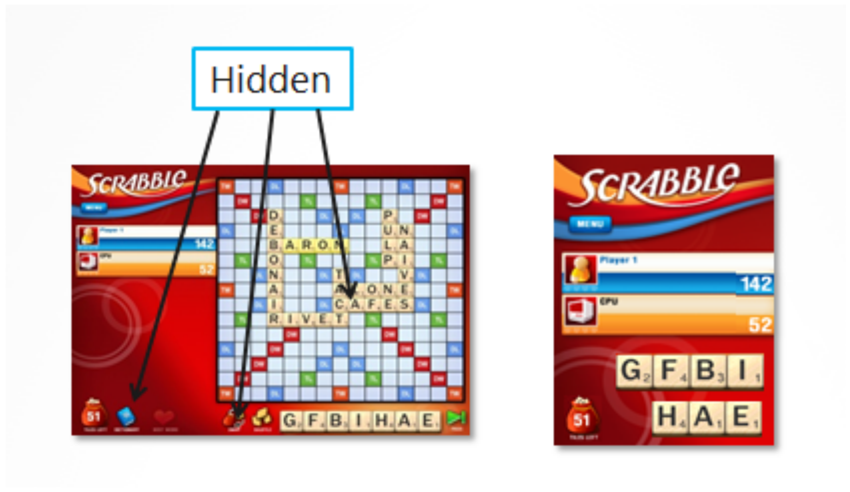
3.7.1.3.1 MANAGE THE TOP LEVEL LAYOUT

The first thing to consider is whether the application hides content for certain orientations or resolutions, or whether it changes its layout significantly for those orientations and resolutions.

In the following example, the application changes its layout when displaying in portrait mode.



In the next example, the application hides information when shown in portrait mode.



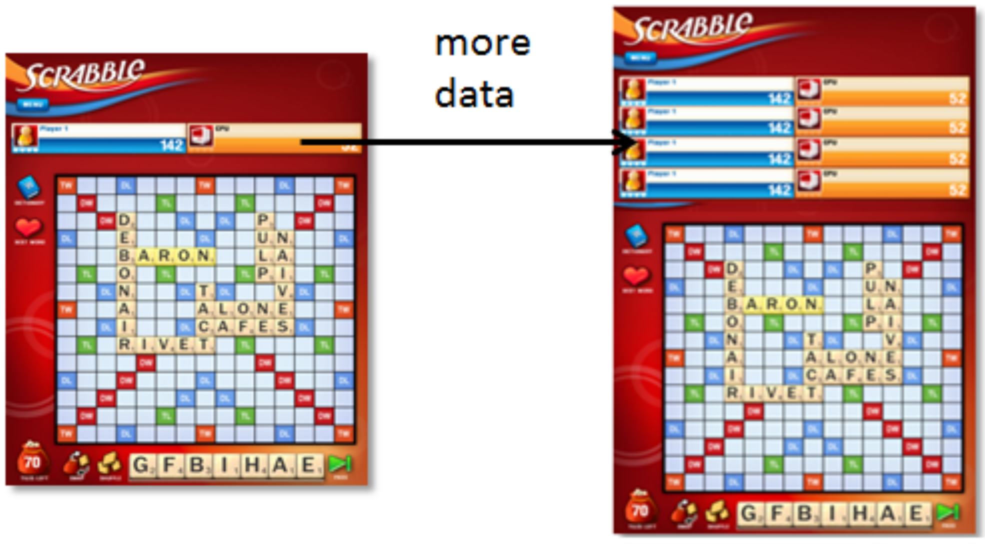
It's a good idea to use Media Queries to handle these sorts of layout changes. Media queries let you query the different dimensions and orientation of available screen real estate. You can use Media Query to specify the layout that best suits the resolution for a specific form factor.

```
// CSS File
@media screen and (orientation: portrait)
{
 /* portrait */
}
@media screen and (orientation: landscape)
{
 /* landscape */
}
@media screen and (max-width: 320)
```

```
{
 /* Docked */
}
@media screen and (min-width: 1024) and (max-width: 1920)
{
 /* Anything that's pretty high res (Slate, Notebook, Desktop) */
}
@media screen and (min-width: 1921)
{
 /* Now we are talking about super high res stuff... */
}
```

3.7.1.3.2 USE GRID TO DEFINE A FLUID LAYOUT

Within each orientation or resolution range, the exact screen dimensions can vary. Rather than simply stretching the current layout, the application can take advantage of extra space by displaying additional data. One way to accomplish this fluid layout is to use the CSS grid and to use "fractional units" (fr) to identify which portion of the application should distribute available space. The CSS grid also enables you distribute content across multiple rows and columns.



By default, Visual Studio and Expression Blend for HTML come with default templates that are built for fluid layout in mind. Here is some HTML to get you started if you do not have access to the templates:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>Layout Testing</title>
    <style type="text/css">
        body, html, .grid { height: 100%; margin:0px; }
        .grid
        {
            display: -ms-grid;
            -ms-grid-columns: 120px 1fr;
            -ms-grid-rows: 100px 1fr 60px;
        }
        .title
        {
           -ms-grid-column: 2;
           -ms-grid-row: 1;
        }
        .content
        {
           -ms-grid-column: 2;
           -ms-grid-row: 2;
        }
    </style>
</head>
<body>
    <div class="grid">
        <div class="title">
        </div>
        <div class="content">
        </div>
    </div>
</body>
</html>
```

3.7.1.3.3 MANAGE SPACING WITHIN YOUR CELLS

You should use the right “fluid aware” elements to manage the available space inside the individual cells of your grid. For example, the Flexbox and the MoCo will automatically arrange and distribute more content.

3.7.1.3.4 RECOMMENDATIONS

- **DO** use media queries to identify the different resolutions that your layout will be specifically targeting.
- **DO** use layout features that adapt to different display sizes.
    - **Consider** using **CSS Grid** as the base layout to define your application regions
    - **Consider** using **CSS Flexbox** to manage space within the grid (aka cells)
- **DO** use built-in controls that are designed for fluid layout and will intrinsically take advantage of any extra screen real estate, such as MoCo.
- **CONSIDER** using CSS to style your controls and page as much as possible—use gradients, rounded corners, and so on.

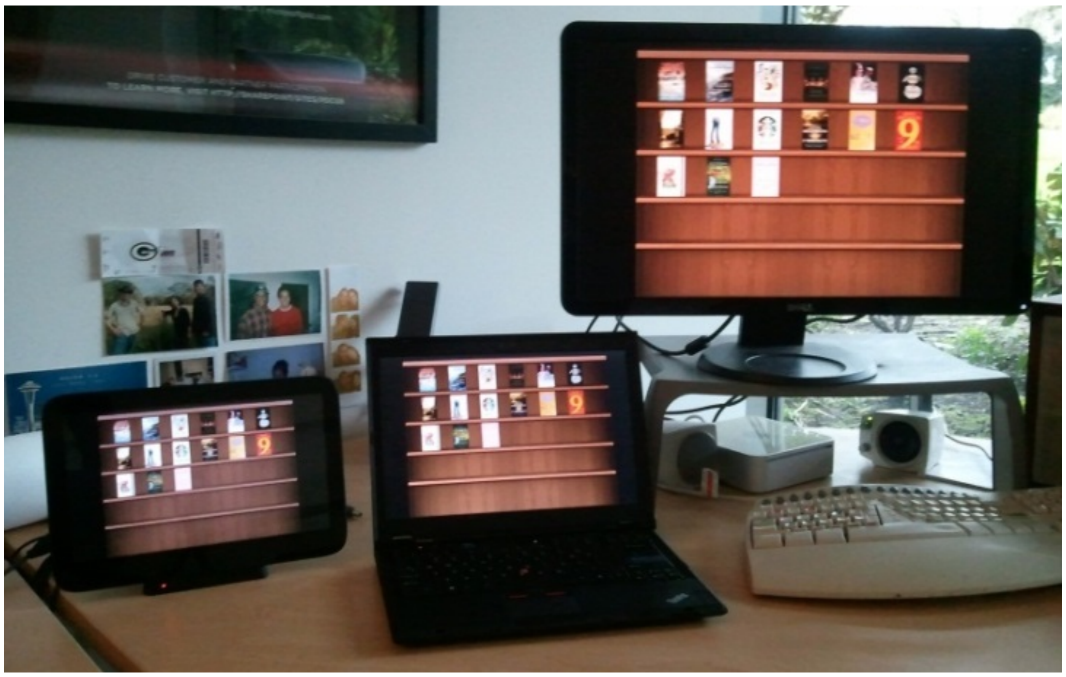
- **CONSIDER** using vector based UI (such as SVG, XAML) for app resources if rendering performance is acceptable.

3.7.1.4 SCALE TO FIT APPS

Some apps, particularly games and game-like rendered UI may not be able to easily take advantage of more space with higher screen resolutions. For these apps we've created a template that allows you to "scale to fit" a layout for an app that was designed for 1366x768. This isn't ideal for all UI because it makes things physically big on desktop monitors but it does allow for these types of UI to still be immersive on different screens without significant work from the developer.

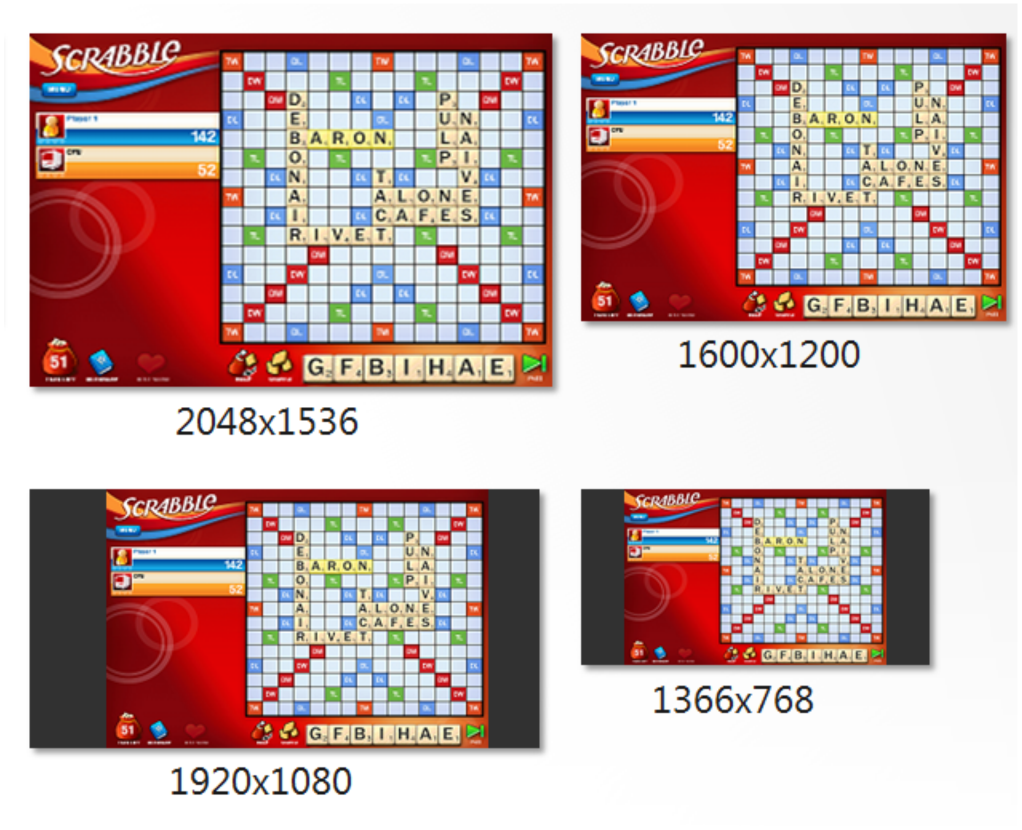
In a scale to fit layout, content and margins remain fixed and are stretched to fit the size and aspect ratio of the screen.

- Provide a quick and easy layout for porting existing web functionality.
- Preserve layout and aspect ratio.
- Can be targeted to a specific form factor.
- Offer letterboxing and image stretching.



3.7.1.4.1 SPECIFY A SCALE TO FIT VIEW

The framework can automatically scale the view size to the available area of the display device. If the aspect ratio does not match, letterboxing will occur. See the following figure for an illustration of how this scaling might appear:



To specify the fixed amount of space to scale your application to fit to, use the Corsica WinJS.UI.ViewBox control which will scale the application to fit the specified dimensions of the content.

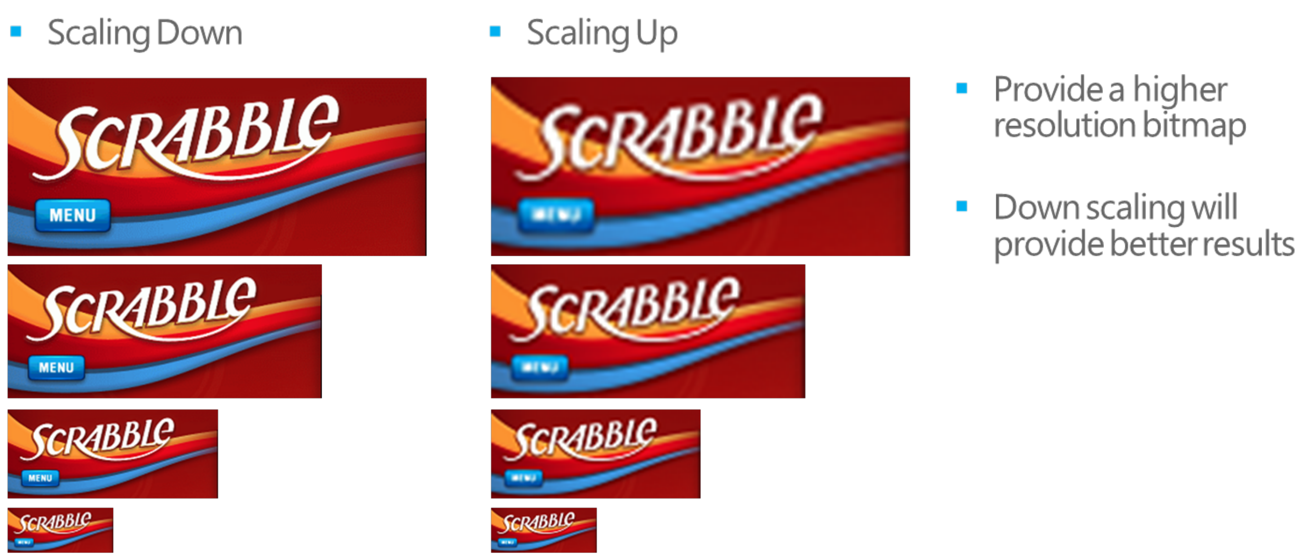
```
<!DOCTYPE html>
<html>
    <head>
        .application {
            width: 1024px;
            height: 768px;
        }
    </head>
    <body>
        <div class="application" data-win-control="WinJS.UI.ViewBox">
            // App layout goes here
        </div>
    </body>
</html>
```

Specify multiple scale to fit views:

```
<!DOCTYPE html>
<html>
    <head>
        .application {
           width: 1366px;
           height: 768px;
        }

        @media screen and (-ms-view-state: snapped) {
            .application {
                width: 320px;
                height: 768px;
            }
        }

        @media screen and (-ms-view-state: full-screen) {
            .application {
                width: 1366 px;
                height: 768px;
            }               }

        @media screen and (-ms-view-state: fill) {
            .application {
                width: 1024px;
                height: 768px;
            }               }

        @media screen and (-ms-view-state: portrait) {
            .application {
                width: 768px;
                height: 1366px;
            }
        }
    </head>
    <body>
        <div class="application" data-win-control="WinJS.UI.ViewBox">
           // App layout goes here
        </div>
    </body>
</html>
```

3.7.1.5 FIXED APPLICATION ASSETS

Developers should ensure to provide images that can be stretched with their layout. Ideally, you should use scalable vector graphics (SVG) and avoid bitmap assets. If bitmap assets are required, ensure that images are at least twice (140%) the size of assets that are required at for the target device. This ensures that images can be scaled to a variety of resolutions without getting blurry. (Scaling large images down yields much better results than scaling smaller images up.)


Scaling Down
Scaling Up
Provide a higher resolution bitmap
Down scaling will provide better results

3.7.1.6 SIZING EVENTS

Notification Scenario	User Action	Event name
Docked in MoBar	Open switch menu, pin app to bar, untethering action	Windows.UI.Immersive.Application.Docked
Undocked from MoBar	Open switch menu, unpin app to bar. Apps may also undock programmatically.	Windows.UI.Immersive.Application.Undocked
MoBar shown	Show bar	OnResize
MoBar hidden	User hide bar	OnResize
Rotation	If accelerometer present and user rotates, button?	Windows.UI.Immersive.Rotation
Soft-keyboard shown	User puts focus in a text field	Windows.UI.Immersive.InputPane.Shown
Soft-keyboard hidden	Focus leaves a text field	Windows.UI.Immersive.InputPane.Hidden
Charm bar shown	User swipe	None
Scale Factor changed	User projects, changes monitor or chooses app setting	OnResize

3.7.1.7 HANDLING PORTRAIT AND ROTATION

Landscape
Portrait

3.7.1.7.1 ROTATION OVERVIEW

Running Windows on a slate and convertible form factors will become increasingly common with the release of Windows 8. These form factors make the user's experience more personal and enjoyable because they are held in the user's hands and can be interacted with touch. Users will be able to rotate these devices easily, changing between landscape and portrait orientations. Switching between these two orientations lets the user choose a layout that best fits their task or posture. Users will expect application layouts that support the portrait orientation from the applications that run on slates. It is crucial that the portrait layout of your application be designed intentionally to fit the needs of your users even if you decide not to support portrait orientation.

Windows portrait point of view

- All applications must support a landscape view and may optionally support portrait for devices that support it.
- Portrait will be used but it won't be as common as landscape.
- Be intentional about your portrait view, even if that means you disable it.

Benefits of supporting portrait orientation

Encouraging ergonomic and comfortable posture
People manipulate objects for personal comfort based on their level of fatigue, their eye sight, available light and many other factors. The ability to rotate a device to portrait adds to this human-interactivity for slates in Windows 8. Supporting a portrait layout for your application on Windows 8 will allow a user to rotate the device based upon the constraints of her surroundings, her posture, and other factors.

Providing a view optimized for vertical tasks
If a task makes use of vertical content, a well-designed portrait layout may be the ideal view. Some tasks that involve vertical content include: reading (e.g. books, webpages, PDFs, email), scrolling long, vertical lists (e.g. scanning an email inbox), or viewing vertically-oriented media (e.g. portrait photographs).

Challenges of supporting portrait orientation

Devices that don't support changing orientation
Consider the devices your application will run on when deciding if you should support portrait orientation. Windows 8 is designed for many devices that cannot change orientation, such as laptops, desktops, family hubs, and others. You may support portrait, but many devices will only see your landscape view.

Windows 8 will run on many devices that don't support portrait

How would the use of widescreen impact your portrait layout?
Because Windows 8 is designed and optimized for widescreen monitors in landscape orientation, consider how your portrait layout could be adjusted if your application were run on a widescreen monitor in portrait orientation. Your available horizontal space will be severely limited.

Does your application have UI elements that can't be moved or modified to fit a portrait layout?
Consider how your portrait layout should present UI elements that cannot be reflowed when changing orientation; some options include (but are not exclusive to) resizing and cropping the element to fit. User-customized UI elements, images, and videos are all examples of elements that cannot be reflowed for portrait layout. User specified UI elements should remain exactly where the user placed them, and the aspect ratios of images or videos could be too wide for portrait orientation (especially if the image or video is widescreen).

3.7.1.7.2 PORTRAIT GUIDELINES

Use the following guidance to help you consider how and when that portrait layout can be used in your application.

Design for landscape orientation first

Your application or feature must support landscape orientation. Windows 8 is optimized for landscape orientation; the best user interface designs should target landscape-based layouts. Design a layout for portrait orientation as a secondary layout to for your application. <u>All applications must support a landscape view they may support portrait but only if the hardware supports it.</u>

Support portrait orientation if the experience of you user in portrait can be as good or better than their experience in landscape

The portrait view must add value to the experience of your user. Give your user a reason to thank you for supporting portrait orientation.

Do not support portrait orientation if the experience of your user is worse in portrait than it is in landscape

Chose to lock your orientation to landscape instead of showing an unsatisfactory or incomplete view with portrait orientation. Ensure that you can be proud of the look, feel, and functionality of your portrait view. Supporting portrait orientation but showing significant letterboxing, cropping, and squishing your UI could make your UI ugly and unusable.

Preserve the current application context when orientation changes

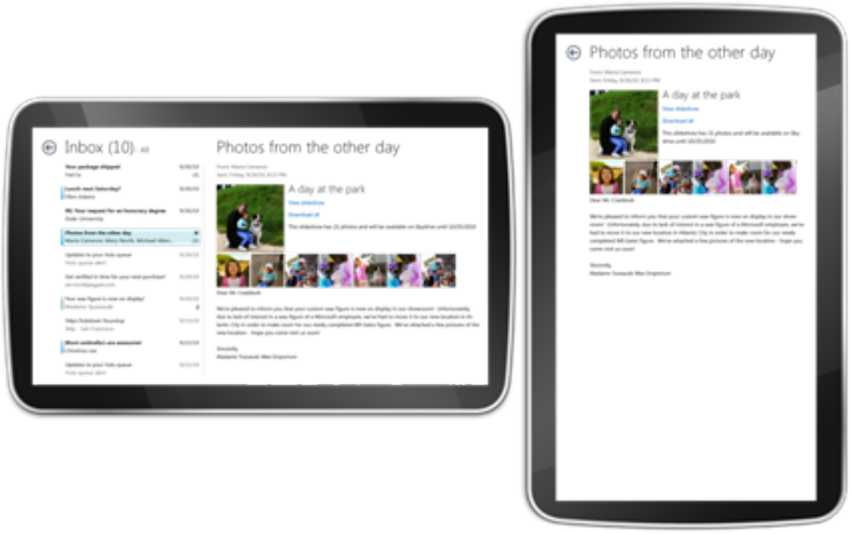
User context is not lost or changed by rotating the device. App state is maintained, entered text is preserved, and selection persists.

Your portrait view should be consistent with your landscape view

The users experience in portrait view should be similar to their experience in landscape view. To ensure the user's experience is consistent, functionality and visual elements should be the same in both portrait and landscape (wherever possible). Portrait view should not contain features that are not available in landscape view or "delighters".

Your portrait and landscape layouts should be optimized for their respective orientations, but they should also look related to one another. Your portrait layout should be a version of your landscape view that has been optimized and refined for portrait orientation,
If your portrait view is too cluttered and busy, try to reduce clutter by removing, grouping, or isolating user interface elements on the application page. If, for example, Your application uses a pane to display content in landscape, it may be appropriate to isolate the content in the pane as a popup in portrait view.

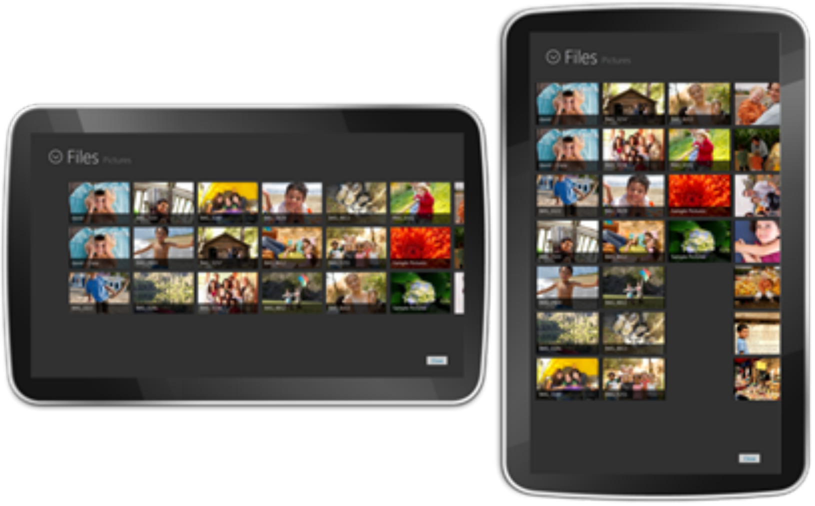
When considering the layout of your portrait view it should be most related to your 4:3 filled landscape view.



Applications should maintain the same overall layout and context between portrait and landscape. It is ok to hide parts of your landscape layout in portrait to focus on the task. In this case, Mail hides its list view but the application still works the same way and context is preserved.

Landscape and portrait views fill the screen

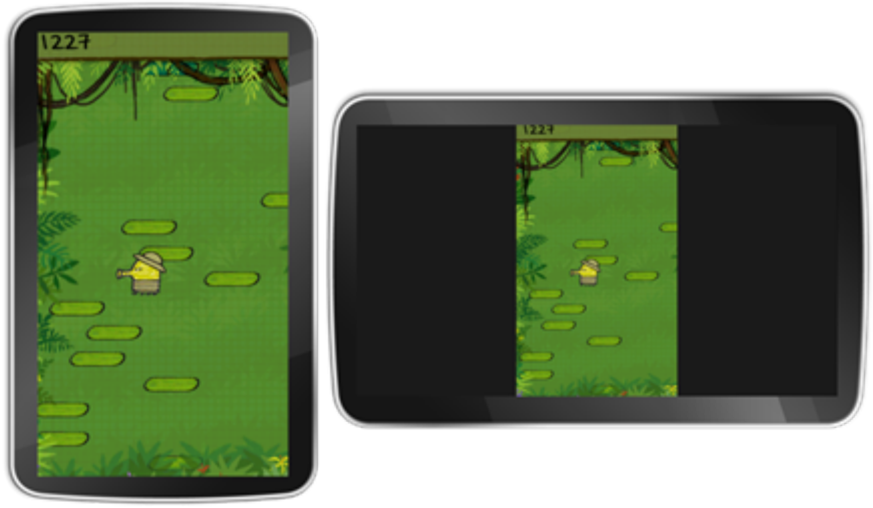
Fill the available screen real estate with content. Where possible, reflow or resize content to avoid leaving blank margins.



In this example the picker reflows to fill the screen. Where possible relayout your content to fit the screen avoiding blank spaces.

Portrait-centric games

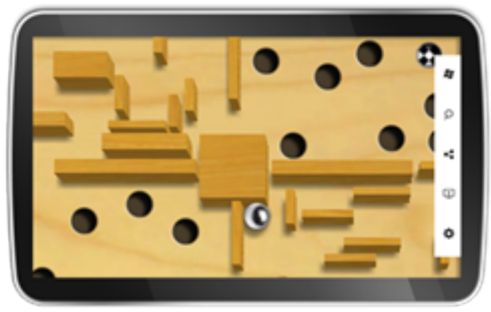
If the focus of your application is the portrait layout, you may default your view to portrait. You must still provide a layout for landscape orientation for devices that do not support orientation. In the absence of a better solution for a portrait-focused application, you may use letterboxing to make your application viewable in landscape orientation for desktops and laptops. *Doodlejump* is a contemporary example for a portrait-centric app. Most applications should be landscape centric.



Even portrait-centric games must support landscape for desktop monitors and laptops that don't support portrait. In cases where content can't reflow show letterboxing.

Orientation independent games

If your application is orientation independent lock the view to the landscape orientation. This will ensure that system UI and elements are shown in predictable places. *Labyrinth* is a contemporary example for an orientation independent app.



Set your preferred orientation to landscape if you are an orientation independent game to ensure that system chrome shows in a consistent location.

Rotation animations

The system provides a rotation animation. Do not attempt to override or create a custom review animation on rotation as it will conflict with the system animation. Locking the system orientation while still rotating the view will yield unexpected results to the user as the edges will not behave predictably.

3.7.1.7.3 IMPLEMENTATION GUIDANCE

Setting initial orientation preference

In the application manifest developers can specify the initial orientation of their application to ensure that the Splash Screen and other system elements are displayed in the right orientation when the application is launched.

```
// package.appmanifest
<VisualElements InitialRotationPreference="landscape">
</VisualElements>
```

Setting application orientation preference

Developers can set an application orientation preference at run-time which will set the orientation of the application.

```
function onLoad() {

    //instantiate the orientation manager object
    var orientation = new Windows.UI.Immersive.DisplayOrientation();

    //read the state
    var state = orientation.autoRotationState;

    //read the orientation capabilities
    var capabilities = orientation.orientationCapabilities

    //set the orientation preference
    capabilities.orientation = "Portrait";
    capabilities.flipEnabled = true;
    orientation.orientationCapabilities = capabilities;
}
```

Responding to orientation changes

Developers can respond with javascript to rotation change events.

```
function onLoad() {
    var dDispProp = Windows.Graphics.Display.DisplayProperties;
    dDispProp.addEventListener("orientationchanged",
    onDisplayPropertiesEvent, false);
}
```

```
// Handle Orientation change
function onDisplayPropertiesEvent ()
{
    var oOrientation =
Windows.Graphics.Display.DisplayProperties.currentOrientation;
}

// When app closes, un-register for Orientation change event
function onClose()
{
    var dDispProp = Windows.Graphics.Display.DisplayProperties;
    dDispProp.removeEventListener("orientationchanged",
    onDisplayPropertiesEvent, false);
}
```

Developers can can also style there application in response to a rotation change event.

```
// CSS: Hide the first grid column in portrait
@media screen and (-ms-view-state: landscape)
{
    .grid
    {
        -ms-grid-columns: 320px auto;
    }
}

@media screen and (-ms-view-state: portrait)
{
    .grid
    {
        -ms-grid-columns: 0px 768px;
    }
}
```

3.7.2 VIEW SCALING

The Windows scaling system provides relatively consistent physical sizes for UI elements on the screen even though pixel density of screens can vary between different hardware. Maintaining physical size is crucial to ensure touch targets, per the touch interaction guidelines. If the pixel density of a particular screen is above a defined threshold, the scaling system applies a scale factor to ensure these physical sizes. There are many system components and frameworks working in concert to ensure that UI scales consistently and predictably, without significant user or developer consideration. Keeping the following design considerations in mind will ensure that applications look great regardless of how they are scaled.

Standard Slate
10.6″ 1366x768
148 PPI

HD Slate
10.6″ 1920x1080
208 PPI

3.7.2.1 SCALE PERCENTAGES

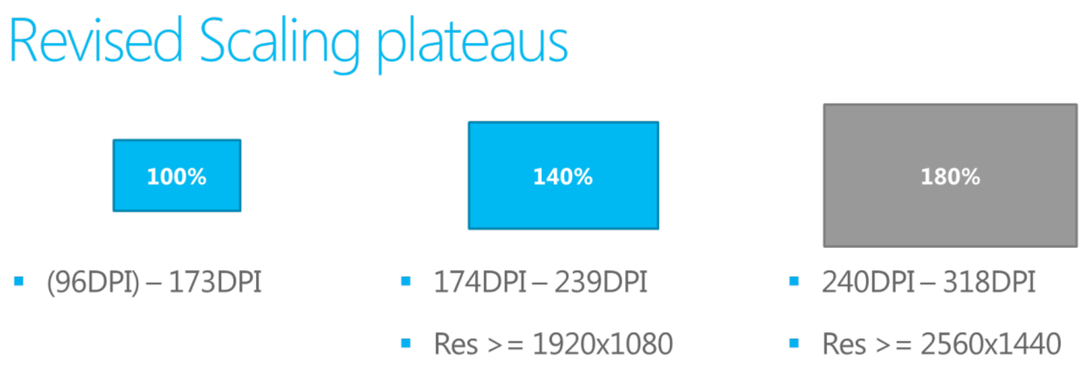
The system defines pre-defined scale percentages that map to a defined range of pixel densities. The scale percentage is selected based upon the system's current pixel density.

How Scale Percentages are Chosen:

The system defines the scale percentages and building a proper fixed or fluid layout will make correctly handle plateaus. The smallest touch target size defines when to switch scale PERCENTAGES. The recommended minimum size is 9mm, but it's possible to provide acceptable targeting down to 7mm due to the touch platform targeting.

3.7.2.2 OUR TARGET DEVICE

Our desgined target device is 10.6" @ 1366x768 = 148PPI and we need to ensure all of our UI looks great on this. This is defined as our 100% scale percentage. The other two percentages make sure that UI is touchable on machines with higher pixel densities.



Revised Scaling plateaus

100%	140%	180%
(96DPI) – 173DPI	174DPI – 239DPI	240DPI – 318DPI
	Res >= 1920x1080	Res >= 2560x1440

3.7.2.3 HANDLING SCALING PERCENTAGES

Use adaptive layouts:

Adaptive layouts ensure that UI looks good and is touchable on many different screens. Additionally it allows for accessibility support, and allows for your application to support various system-level states. See Adaptive Layout for more detail.

File setup & guidelines:

- All assets should be produced **@96dpi**
- Use the **20px grid system** in your layouts and type sizes that map to it well.

Recommended default type size:

The recommended default type size is **11pt** to ensure comfortable reading and reliable localization. 9pt is the smallest legible size but can be problematic. It should only be considered in non-essential situations. Use Segoe UI (not Segoe UI Light, Segoe UI Semibold or other new fonts) on type sizes smaller than 11pt as there is no hinting for these fonts.

3.7.2.4 SCALABLE APPLICATION ASSETS

Applications and their images will be scaled up on higher DPI machines to match the plateau. Images and other assets should scale without displaying artifacts or blurriness. Ideally, provide vector images. CSS primitives, SVG, and XAML assets can be scaled without any extra work. When bitmap assets are required, provided three versions of each asset, one that corresponds to each plateau.

Ensure to specify widths and heights on images loaded by the resource loader in order to allow for a consistent layout.

3.7.2.4.1 USE THE RESOURCE LOADER

For local and web assets the resource loader retrieves the bitmap asset that corresponds to the current scale percentage.

This example specifies assets for each plateau with different names, all in the same folder.

```
foo.scale-100.jpg
foo.scale-140.jpg
foo.scale-180.jpg
```

Developers may also specify assets for different plateaus in different folders.

```
…\images\scale-100\foo.jpg
        \scale-140\foo.jpg
        \scale-180\foo.jpg
```

3.8 RESOURCES AND LOCALIZATION

Windows is used worldwide, on a variety of different machines and form factors. It is vital to design applications so that resources, such as strings and images, are separated from their code. This lets them be easily localized, change for different scaling factors and accessibility options and a myriad of other user and machine contexts.

You should separate resources from code, and use the new resource infrastructure in Windows 8 to handle the selection of the most appropriate resources to best match a particular user's runtime environment. This enables:

- Maintenance of the code separately from the content in the resources.
- Creation of localized and tailored variants of each resource, and the ability to add new variants later.
- Localization of resource content separately from the code, by means ranging from language and culture experts to machine translation as appropriate.
- The display of different resources based on different configurations and user settings.

3.8.1 QUICK START

3.8.1.1 LOCALIZATION

1. Copy all strings from code and/or markup into a resource file (ResW or ResJSON), providing resource comments as necessary.
2. Add resource references in the code and markup to refer to the resource

3. Isolate localizable resource files (e.g. images containing localizable strings, or that should be mirrored LTR/RTL) from language-neutral files, placing them in language tag named folders (e.g. fr-FR/logo.jpg)
4. If localizing HTML, add Res.js to the document and make a call to Win.Resources.processAll() on DOMContentLoaded
5. Be sure to create a PRI file (resources.pri in the root of the package) using MakePRI.exe

3.8.1.2 SCALE FACTOR

1. Create two copies of each image
2. Original size for a typical 92dpi device
3. 150% the size, (e.g. 100x100px image should also have a 150x150px image version)
4. Name the two images <name>.scale-100.<ext> and <name>.scale-150.<ext> and place them side by side in the same folder. Where <name>.<ext> is the name of the image as referenced in code/markup. (e.g. <img src="logo.jpg" /> and the package contains logo.scale-100.jpg and logo.scale-150.jpg.
5. Be sure to create a PRI file (resources.pri in the root of the package) using MakePRI.exe

3.8.2 CREATING RESOURCES

The following sections describe how to localize strings, images, and files.

3.8.2.1 STRINGS

String resources should be stored in separate string tables.

3.8.2.1.1 MICROSOFT APP DEVELOPERS & JUPITER APPLICATIONS

Store strings within a ResW file. A ResW file is simply a .NET ResX file containing only strings.

Note: The following tooling experience may not be available in particular versions of VS or may be incorrect. Follow the manual file format below to author strings.

1. Right click the Project. Select Add Item > Folder. Create a folder named with the correct language tag (BCP47).
2. Right click the new folder. Select Add Item > Resources (ResW) to create a resw file named **resources.resw**.
3. Use the built in designer to add string resources and comments.

Be sure to name the file **resources.resw** unless you know how to explicitly refer to resources in other ResW files (see later sections).

For example, the file **en-US/resources.resw** would be formatted as follows:

```
<root>
    <data name="String1">
            <value>Hello World</value>
            <comment>A welcome message</comment>
    </data>
    <data …
</root>
```

3.8.2.1.2 THIRD-PARTY WWA DEVELOPERS & SAMPLE WWA DEVELOPERS

Store strings within a ResJSON file. A ResJSON file is a basic text file containing a JSON (http://json.org) formatted object.

Note: The following tooling experience may not be available in particular versions of VS or may be incorrect. Follow the manual file format below to author strings.

1. Right click the Project. Select Add Item > Folder. Create a folder named with the correct language tag (BCP47).
2. Right click the new folder. Select Add Item > JSON Resource to create a resjson file named **resources.resjson**.

Be sure to name the file **resources.resjson** unless you know how to explicitly refer to resources in other ResJSON files (see later sections).

For example, the file **en-US/resources.resjson** would be formatted as follows:

```
{
    "String1"               :       "Hello World",
    "_String1.comment"      :       "A welcome greeting",

    "String2"               :       "Something else",
    "_String2.comment"      :       "This is a comment to the localizer"
}
```

Ignore the deeper nested structure VS auto-generates, create your files like the example above.

3.8.2.2 IMAGES AND FILES

XAML, JPG, PNG, other image formats, CSS, HTML Fragments and any other file types are inherently resources, which are typically files by themselves and therefore are already separated from the application code.

All file assets should be marked with qualifiers, which specify when they are appropriate for use in the application UI. This is done by placing them in correctly named folders and/or filenames.

Qualifier Name	Value	Details
Language	**Foldername:** <Any BCP47 language tag> **Filename:** lang-<Any BCP47 language tag>	Specifies the language of the resource. Note that filenames require lang- as a prefix to the language tag.
Scale*	scale-100 scale-150 scale-200	Specifies the scale factor of the image (typically used for images when the application is zoomed or viewed on a higher dpi device.
Contrast	contrast-standard contrast-high contrast-bl contrast-wh	Specifies the contrast theme setting of the system. Standard is when all high contrast themes are off. High is set when in any high contrast mode. Bl represents when Black background/white foreground images are preferred. Wh represents when white background/black foreground is preferred.
Home Region	Homeregion-<Region Code> Any valid **iso-3166-1 alpha 2** two-letter region code, or any valid **iso-3166-1 numeric** three-digit geographic code (identical to the United Nations Statistic Division M49 region codes)	Represents the home location of the user. Typically specified at Windows installation and accessible from the Control Panel. Commonly overridden if some other data provider is known.

*Expect upcoming changes to scale factors pre-PDC.

3.8.2.2.1 EXAMPLES

```
Images/
    Logo.scale-100.png
    Logo.scale-150.png
fr-FR/
```

```
    menu.png
    map.contrast-bl_scale-100.png
```

The above example specifies images to be used within an application. The map resource likely contains French text, is useful in a high contrast theme and is sized appropriately for a scale factor of 100%.

Application developers may also choose to use either folder name or file name, or a mix of the two strategies, to specify assets for a given set of dimensions. For example, the following indicates a file that is localized for US English, is intended for a black-background high contrast theme, and is sized for the 100% scale factor.

```
en-US/contrast-bl/logo.scale-100.png
```

Note: Microsoft Inbox App Developers are impacted by an additional restriction imposed by the build system. Localizable files must be contained in a folder specifying the language at the root of the project. For example, **myproject/en-US/images/logo.png** would be a valid path whereas **myproject/images/en-US/logo.png** would not be.

Additional example:

```
en-US/
    resources.resw
images/
    logo.scale-100.png
    logo.scale-150.png
```

3.8.3 BUILDING

Every package should contain a special file that is an index of all resources in the application. This file will be created at build time and is referred to as a **Package Resource Index file** or **PRI**. The PRI file is packaged with the rest of the application and is used to resolve resources at runtime.

- The PRI file is a binary file stored in the root of the package and named **resources.pri**

PRI files are specifically designed to be the replacement for resources contained within DLLs.

Until VS has integrated directly with MakePRI.exe in PDC-4, application developers will need to generate a PRI file manually. Each package should contain a single PRI file (**resources.pri** in the root of the package).

Note: MakePRI.exe is available in \\winbuilds\release\winmain\XXX\bin\idw\MakePRI.exe

1. Run

```
MakePRI.exe new –pr C:\MyPackage\ -of C:\MyPackage\resources.pri
```

where -pr (project root) is set to the project's source location.

2. Add the PRI to the project Add Item > Existing Item.
3. Make the PRI an Item Type of "Content" by right clicking the file in the Solution Explorer > Properties and choosing "Content" for the Package Action.

The PRI should get added to the package when F5 is run.

Remember to name the PRI file **resources.pri** and place it in the root of the package.

3.8.4 USING RESOURCES

- The PRI is an index of *named resources,* which is the abstract notion of a resource (for example, displayStringForManifest).
- Each named resource has one or more *resource candidates*, where a candidate is the concrete value for the resource and the conditions for which it is appropriate (for example, "Hello World" is a concrete value and is useful when the user's language matches en; "Hola Mundo" is another concrete value for the same item, but it applies to different runtime contexts: it is useful when the user's language matches es).
- For Windows 8, instance values are either string resources themselves or file paths that reference a loose file resource.
- Each resource candidate is stored with what are known as *qualifiers*. Qualifiers describe when a resource candidate is appropriate for use (what the scale factor should be, the language the string is in, etc).

3.8.4.1 HTML

Referring to resources in HTML markup is similar to Corsica JS Toolkit's data-binding for HTML. HTML elements should have an additional **data-win-res** attribute specifying the properties of the element and the names of the resource they should be replaced by.

```
<input type="text" placeholder="Click here" title="Tooltip Info" data-win-
res="placeholder:Placeholder1; title: Title1"/>
<ul>
    <li data-win-res="innerText:ListElement1">Male</li>
    <li data-win-res="innerText:ListElement2">Female</li>
</ul>
```

Where the resource file **resources.resjson** contains:

```
{
"Placeholder1"     : "Click here",
"Title1"           : "Tooltip Info",
"ListElement1"     : "Male",
"ListElement2"     : "Female"
}
```

In order to process these data-win-res attributes, the document must include a JS toolkit called **Res.js**, which exposes a process call. It is recommended that this process call be one of the first calls made after the DOMContentLoaded event, as it critical to the document's UI displaying properly.

```
<script type="text/javascript" src="ms-wwa://Win8UI/Res.js" />

<script type="text/javascript">
    document.addEventListener("DOMContentLoaded", function() {
            Win.Resources.processAll();
    }, false);
</script>
```

Note: Res.js is available at \\winbuids\release\winmain\XXX\bin\Win8UI\js\res.js

Where possible, bind a resource to the textContent property instead of innerHTML to replace strings that don't include their own markup. The textContent property is much faster to replace than innerHTML.

3.8.4.2 XAML

Not currently available (coming soon).

3.8.4.3 JS, C#, C++

When accessing resources in Js, C# or C++ code instead of using hard-coded text, the Windows.ApplicationModel.Resources WinRT APIs should be used. So instead of writing the following in JavaScript:

```
alert("Hello World");
```

you should use:

```
var R = new Windows.ApplicationModel.Resources.ResourceLoader();
alert(R.getString("HelloWorld"));
```

where HelloWorld is the name of the string "Hello World" in the resource.resw or resource.resjson file.

In C#, this would be achieved using:

```
ResourceLoader R = new
Windows.ApplicationModel.Resources.ResourceLoader();
String Hello = R.GetString("HelloWorld");
```

3.8.5 HTML FRAGMENT LOADING

When placing resource references in HTML fragments, the data-win-res markup must get processed before injecting the fragment into the current document. Therefore, it is recommended that the process call be made in the data-win-fragmentLoad function.

```
<script type="text/javascript" data-win-fragmentLoad="init" >
    function init(elem){
            Win.Resources.processAll(elem);
    }
</script>
```

3.8.6 HTML DATA-BINDING AND RESOURCES

When using HTML data-binding and resources, it is important to process resources prior to data-binding. For example, suppose we have a string

```
You have {count} messages
```

As with best localization practices, the string should be kept as a complete sentence in the resource string and may contain HTML markup for the binding.

```
<script type="text/javascript">
    document.addEventListener("DOMContentLoaded", function(){
    Win.Resources.processAll();
    Win.Binding.processAll();
}, false);
</script>

…

<span data-win-res="innerHTML: messageCount">
You have <span data-win-bind="innerText: count"></span> messages
</span>
```

```
{
    "messageCount" : "You have <span data-win-bind="innerText:
count"></span> messages",
    …
}
```

3.8.7 APPLICATION MANIFEST

Manifest localization is still coming online, until it is ready some strings may not resolve to resources.

All displayable strings and icons in the manifest are localizable. String references can be done by putting a resource reference in the place of a hardcoded string.

```
<DisplayName>ms-resource:String1</DisplayName>
```

For example, the above example refers to a string called “String1” in resources.resjson or resources.resw.

File references stay the same as they refer to the logical file (logo.jpg) which maps to physical file resources (logo.scale-1x.jpg)

```
<VisualElements Logo="logo.jpg" … >
```

3.8.8 LIBRARIES, FRAMEWORKS AND OTHER COMPONENTS

Applications often take dependencies on or reference framework packages, .Net portable libraries, class libraries, control libraries, or have multiple components. Resources for each component are separately managed and accessed.

Programmatically, components can access their own resources with their own ResourceLoader, by passing the component’s name. The component’s name is the framework’s package identity name, the class libraries assembly name or the .Net portable library’s simple name.

```
ResourceLoader R = new
Windows.ApplicationModel.Resources.ResourceLoader("Win8UI");
R.GetString("loadingStr");
```

HTML references to resources that aren’t the default main application must specify more explicitly the component’s resource using a resource URI.

```
<span data-win-res="innerText: ms-
resource://Win8UI/Resources/replyAll">Reply All</span>
```

3.8.9 OVERRIDING THE CURRENT CONTEXT

In some particular cases, being explicit about the language, scale or other context qualifier when loading resources may be helpful. This can be done by overriding a context object and passing that into the advanced resource loading APIs

```
var R =
Windows.ApplicationModel.Resources.Core.ResourceManager.current.mainResour
ceMap.getSubtree("Resources");

var context = new
Windows.ApplicationModel.Resources.Core.ResourceContext();
context.language = "fr-FR";
alert(R.getValue("String1", context));
```

Advanced resource loading APIs are under the Windows.ApplicationModel.Resources.Core namespace, where the singleton Resource Manager at ResourceManager.Current manages all the various ResourceMaps for each of the various components (framework dependencies, libraries, etc).

Within a ResourceMap, a hierarchy of file and other resources can be accessed by getting a sub ResourceMap. To access string resources the sub ResourceMap is the filename of the resource file it was stored in (i.e. resources for strings within resources.resjson).

3.8.10 USING MULTIPLE RESOURCE FILES

Some applications may want to separate resources into more than just the default (resources.<ext>) resource file, in order to ease maintenance and provide separation of components. Resource URIs can be used to access resource files more explicitly.

```
<span data-win-res="innerText: ms-resource:/Errors/AlreadyRegistered">
    User has already been registered
</span>

<script>
    var R = new Windows.ApplicationModel.Resources.ResourceLoader();
    alert(R.getString("ms-resource:/Errors/OutOfMemory"));
</script>

<DisplayName>ms-resource:/Manifest/DisplayName</DisplayName>
```

3.8.11 REFERENCES

1. Application Resource and Localization

4 ELEMENTS OF A MOSH APP

4.1 MODERN ACTIVATION

Modern activation refers to the process of preparing an instance of a Modern Application to presnt to the user. This is done by using the activation registration information specified in the Package Manifest for the launch scenario. This registration information can either be specified as an ActivatableClassID or an HTML page to navigate to. See the **Error! Reference source not found.** section or the activation sections that follow for more information about registration.

There are two types of application instances that can be created by the system: Stateful Immersive Apps and Stateless Hosted Versions of the Apps.

Stateful Immersive Apps have the following characteristics:

- They appear as a Snapped App or full screen in the MoBody.
- They never appear in both of these places at the same time.
- They remember their state when restarted so users feel as though the application never closed.
- They show up in the Switch List.

Stateless Hosted Apps, which are hosted by an Immersive App, have the following characteristics:

- They render their UI within the system flow's window. For example, the Share Flyout and various Pickers demonstrate this characteristic.
- They never appear in the Snapped App View—they can't be docked.
- They share the same data with the Immersive App, such as authenticated state, content stores, and so on.
- They do not navigate or manipulate the view of the hosting Immersive App.
- They do not change the state of the hosting Immersive App.
- They are used for short, directed tasks.
- Their code is focused solely on the short, directed task.
- They do not show up in the Switch List.

The type of application instance created by the system varies based on the launch scenario the system is trying to accomplish. The following is the complete list of system launch scenarios and the accompanying type of application instance that is created.

Launch Scenario	Stateful Immersive App	Stateless Host App
Tile of Non Running App	X	
Switch to Non Running App	X	
Secondary Tile (Content Tile)	X	
Search	X	

Share		X
Toast	X	
File Association	X	
Protocol Handler	X	
File Picker Extension		X
Send		X

During activation, if the application isn't instantly ready the system will display the application's splash screen to give the user an indication that the application is loading. Once the application has completed the activation process, its window will replace the application splash screen.

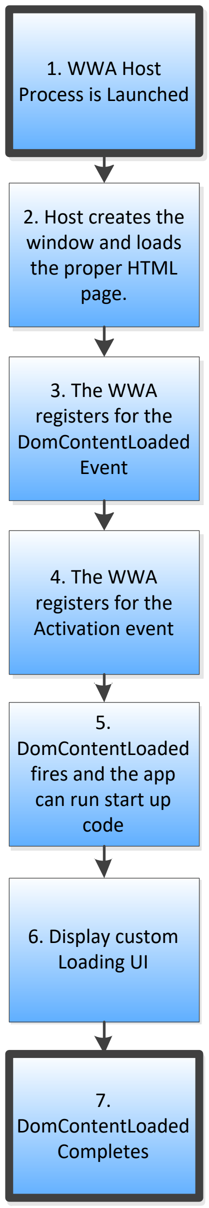
4.1.1 WWA ACTIVATION

WWA activation refers to the process of invoking a Windows Web Application in the context of one of the modern activation contracts. A WWA can register for these contracts in its manifest. By doing so, it asserts that it provides the correct activation behavior for each contract. For example, a WWA which registers for the File activation contract must, when activated to open a file, open the file it is given as part of the activation.

WWA activation consists of two distinct steps: launching and activation. The diagrams that follow illustrate the flow of each step.

4.1.1.1 LAUNCHING A WWA

A WWA undergoes the launch process as part of activation whenever it was previously shut down. A shutdown could have been caused by PLM, a system shutdown, the app itself, or the user.

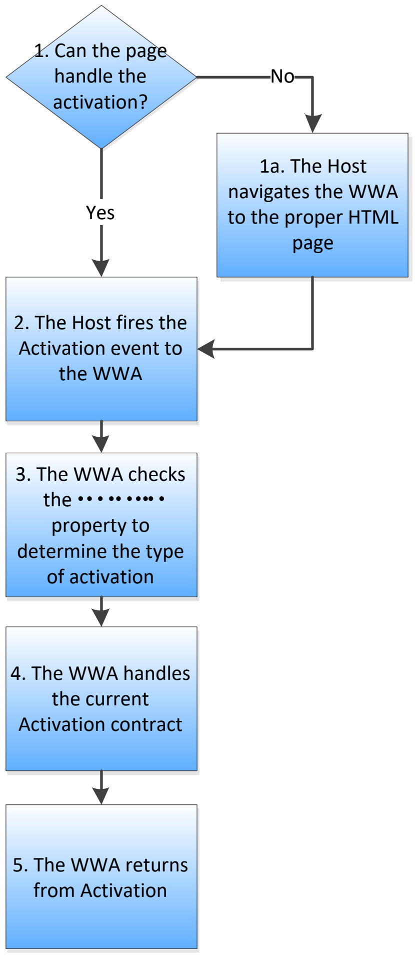

1. WWA Host Process is Launched
2. Host creates the window and loads the proper HTML page.
3. The WWA registers for the DomContentLoaded Event
4. The WWA registers for the Activation event
5. DomContentLoaded fires and the app can run start up code
6. Display custom Loading UI
7. DomContentLoaded Completes

Launch Steps:

1. Modern activation launches a new WWA Host process in the correct Application Container with the App ID of the application being launched. A splash screen is shown while the application is loading.
2. The WWA Host creates the correct type of window for the contract it is being activated for and navigates to the proper HTML page. The WWA must declare this 'StartPage' in its package manifest. It can be declared once for all contracts or on a per-contract basis.
3. The developer can then register for the **DomContentLoaded** event. This will be the first event that the developer receives where they can begin to manipulate the DOM and initialize their application.
4. The WWA can also register an event listener for the **Activated** event.
5. Once the HTML DOM has been fully loaded **DomContentLoaded** fires and the WWA's handler is invoked. Here the WWA can run any start up code that their app requires that is independent from the exact activation they are being launched for.
6. At this point the developer may also display their own loading UI. This UI will be the first thing the user sees after the splash screen is torn down. Any long running tasks should be done when this UI is visible instead of being done during **DomContentLoaded** or **Activated** when the splash screen is still visible.
7. Once the app's event handler returns and **DomContentLoaded** completes the **Activation** event will be fired to the app.

4.1.1.2 ACTIVATION

Activation can happen as part of the initial launch of the app or when it is already running (aka Activation will happen during the initial launch of the app following the completion of **DomContentLoaded**, but before the **onLoad** event is fired. It can also occur after the first launch while the app is already running (aka. reactivation).


1. Can the page handle the activation?
No
Yes
1a. The Host navigates the WWA to the proper HTML page
2. The Host fires the Activation event to the WWA
3. The WWA checks the •• • •• ••• • property to determine the type of activation
4. The WWA handles the current Activation contract
5. The WWA returns from Activation

Activation Steps:

1. The WWA Host checks to see if the current page is the page registered to handle the activation in the package manifest. If it is not then the Host will navigate the app to the start page specified in the manifest automatically. This page must be a local HTML file that was installed with the package.
    a. The new page is loaded and the Activating flow repeats allowing the app to handle the Activation contract on the new page.
2. The Host fires the **Activation** event to the WWA, invoking the handler that was registered during the launching step. If the app was already running this happens immediately. If the app is being launched this happens once **DomContentLoaded** has completed.
3. Within its Activation Handler the WWA can check the *contractId* property on the event arguments to determine which contract it is being activated for. It should then run its activation code for that contract.
4. The WWA will extract out the contract parameters and load the appropriate experience. It can also check the *reason* property on the event args to see if it should restore state. If *reason* is set to *ResumeFromTerminated* the app should load any state that it has saved that is applicable to the content being loaded.
5. Once the WWA has returned from their **Activation** event handler, the WWA Host will continue execution of the app. If activation was part of a launch, the splash screen is hidden and the **onLoad** event fires. If the app was already running, execution continues normally.

4.1.1.3 TYPES OF WWA ACTIVATION CONTRACTS

There are two primary types of activation contracts as far as WWAs are concerned. The first are single window activation contracts. The second are multi-window activation contracts.

4.1.1.3.1 SINGLE WINDOW CONTRACTS (STATEFUL IMMERSIVE APPS)

Some examples of single window contracts are Tiles, Search, and File Associations. These contracts require the WWA to fulfill the contract in its main window. This means that all activations of this type are sent to a single window. There is no concept of multiple main windows in a WWA.

To handle Single Window activations, WWAs should:

- Declare a single HTML page in their manifest that can handle all the main window activation contracts that it registers for.
- Ensure that their activation handler can accept activations for all contracts that the app is registered for at any time while the app is running.
- Preserve the user's current state when the app is shut down and restore that state when the application is launched.
- Preserve any important user data when they switch to a new activation context. This might mean showing the new activation context within the existing view, creating a tabbed view, or temporarily saving the user's data.

4.1.1.3.2 MULTI-WINDOW CONTRACTS (STATELESS HOSTED APPS)

Two examples of multi-window contracts are App-to-App Picking and Share. These contracts specify that the WWA must fulfill the contract in a new, temporary window. This means that only one activation is sent to the application when it is invoked for these contracts. Each multi-window activation creates a new window, loads a new HTML page, and fires a new **Activation** event to that page.

To handle a multi-window contract, a WWA should:

1. Declare an HTML page in their manifest that is specialized to handle the contract the app is registering for.
2. Structure the activation handler to only handle a single type of contract. This handler will only be called once in the lifetime of the app. Reactivations are not possible.
3. Avoid restoring/saving state.   These types of activations are meant for temporary flows. If the user dismisses the flow they are explicitly saying they are done with it and therefore the state of the flow should never be brought back.

4.1.1.4 WINDOWS.UI.WEBUI.WEBUIAPPLICATION API DETAILS

Windows.UI.WebUI.WebUIApplication API

Event	Description
Activated	Occurs whenever the WWA is activated by the system. It is always raised on first launch and can occur while the app is running in single window scenarios. It has the following parameters: • *eventArgs:* An object that contains the parameters being passed to the WWA by the caller. Every contract defines its own unique *eventArgs* object. Please see the sections related to the specific activation contracts for more details on their *eventArgs* objects.

4.1.1.5 SAMPLE JAVASCRIPT CODE

This example shows how single window tile, search, and file contracts can be handled on a single page.

```
<!-- Index.html -->

<script type="text/javascript">

function DOMContentLoadedHandler()
{
    // Logic for main window launches
    // Load State
    // Build basic chrome
    // Show custom Loading UI if necessary
}

document.addEventListener("DOMContentLoaded", DOMContentLoadedHandler,
false);

function activatedHandler(eventArgs)
{
    //display custom loading screen that will show once the splash screen
is torn down

    switch (eventArgs.contractId)
    {
         case "Windows.Launch":
              var tileArguments = eventArgs.token;
               //use the tile parameters to set up the page
               break;
         case "Windows.File":
              var filelist = eventArgs.files;
              //open the file/s in fileList and setup the page to view them
              break;
         case "Windows.Search":
              //initialize the search object and set up the page for search
              break;
    }
}

Windows.UI.WebUI.WebUIApplication.addEventListener("activated",
activatedHandler, false);

</script>
```

This example shows how a multi-window activation for sharing can be handled on a separate HTML page.

```
<!-- Share.html -->
<script type="text/javascript">
```

```
function DOMContentLoadedHandler()
{
    // Logic for Sharing launches
    // Load Sharing chrome
}

document.addEventListener("DOMContentLoaded", DOMContentLoadedHandler,
false);

function activatedHandler(eventArgs)
{
    // display custom loading screen that will show once the splash screen
is torn down

    if (eventArgs.contractId == "Windows.ShareTarget")
    {
         //set up the page to share the data
        var sharedDataPackage =
eventArgs.shareTargetActivationContext.data;
    }
}

Windows.UI.WebUI.WebUIApplication.addEventListener("activated",
activatedHandler, false);

</script>
```

4.1.1.6 REFERENCES

1. WWA Activation Functional Spec
2. WWA Activation Dec Spec
3. WWA Activation API Doc
4. WWA Activation Samples

4.1.2 WINRT UI ACTIVATION

4.1.2.1 OVERVIEW OF WINRT UI ACTIVATION

WinRT UI apps are compiled as unique executable files. These executable files are packaged (along with other assets) into AppX packages. WinRT UI apps register for contracts by declaring WinRT components that are implemented in their exe server via the Application Manifest. When a WinRT UI executable process is started through activation, the app code in the executable calls into the WinRT UI Runtime and delegates the work of activation to WinRT UI Runtime code.

When the activation call occurs, the CoreApplication is spun up , unless already running, and based on a set of heuristic either creates a new thread and new Window corresponding to that thread or re-uses an existing thread & Window. Based on the type of activation, the CoreApplication then forwards a call to the WinRT UI Application Object. This activation calls a method in the WinRT UI application user code that corresponds to the particular type of activation for a given contract. The app is responsible for handling the contract ID/object, creating a visual tree, and assigning this visual tree to the Content property of the Window. The WinRT UI Runtime parsers the XAML for the tree and renders the UI to the screen.

4.1.2.2 SEQUENCE

This is the basic sequence for WinRT UI app activation when clicking on a tile:

1. The OS (shell component) handles a tile click. The tile has an associated a pair of values, appId and contractID. The shell looks up the appID, contractID pair from a catalog and resolves it to a WinRT ACID. This is the app's ACID mentioned in the preceding section.
2. The shell calls **ActivateInstance** for the ACID. As explained previously, the app's executable file is registered with WinRT as an out-of-process server for the ACID. WinRT creates a new process by calling **CreateProcess** for the app's executable file.
3. The app's executable file starts and calls into the WinRT UI Runtime to register its ACID. WinRT UI performs the out-of-process WinRT server work of registering the app's ACID.
4. At this point the application's Winmain function is invoked and Run is called on the WinRT UI Application Object. This in turn invokes the CoreApplication object, which RegistersForActivationFactory an internal class that implements IActivatableApplication. The CoreApplication object is responsible for waiting on the Shell to return that particular class an invoke the Activate method.
5. The Shell calls **IActivatableApplication::Activate** on the CoreApplication's class. The CoreApplication implementation first determins if a new thread and Window need to be created based on the application, then follow the correct semantics determined by the contract mechanics. Based on this, the CoreApplication creates a WinRT UI ViewProvider which is the communication mechanism between the CoreApplication object and WinRT UI.
6. CoreApplication invokes Activate on the ViewProvider which in turn calls a method on the Application object in user code corresponding to the particular activation contract.
7. The user's code implements an overridable method in App.xaml.cs and determines which UI to display. This typically involves creating a UserControl and assigning that UI to the Content property of the Window.

4.1.2.3 WINDOW MANAGEMENT

As with WWAs, WinRT UI represents single window contracts as well as multiple window contracts.

4.1.2.3.1 SINGLE WINDOW CONTRACTS

Some examples of single window contracts are Tiles, Search, and File Associations. These contracts require the application to reuse the existing main window.

To handle Single Window activations, applications should:

- Ensure their application code and manifest opts in to support all the desired contracts
- Set the Content of the Window to the desired UI based on that activation type
- Preserve the user's current state via the Suspend event when that application receives that event

4.1.2.3.2 MULTIPLE-WINDOW CONTRACTS

Some examples of multiple window contracts are App2App Picker, Share and Settings. These contracts require the shell to create a new thread and Window associated with that thread, as well as set the appropriate Content for that particular Window.

To handle Single Window activations, applications should:

- Ensure their application code and manifest opts in to support all the desired contracts
- Set the Content of the Window to the desired UI based on that activation type
- Preserve the user's current state via the Suspend event when that application receives that event

4.1.2.4 WINRT UI API FOR ACTIVATION

Activation requires the application author to use the IApplication and IWindow interfaces to handle activation and set Content to the given Window.

```
public class Application
{
    // Events
    public event EventHandler Exiting;
    public event EventHandler Resuming;
    public event EventHandler Suspending;
    public event UnhandledErrorEventHandler UnhandledError;

    // Methods
    public void Exit();
    public static void LoadComponent([In] object component, [In] string
resourceLocator);
    protected virtual void OnActivated(IActivatedEventArgs args);
    protected virtual void OnFileActivated(FileActivatedEventArgs args);
    protected virtual void
OnFilePickerActivated(FilePickerActivatedEventArgs args);
    protected virtual void OnInitialize();
    protected virtual void OnLaunched([In] LaunchActivatedEventArgs args);
    protected virtual void OnSearchActivated([In] SearchActivatedEventArgs
args);
    protected virtual void OnSharingTargetActivated(
ShareTargetActivatedEventArgs args);
    public void Run();

    // Properties
    public static Application Current
    public ResourceDictionary Resources
}
```

Much of the tooling associated with WinRT UI will create all the template and generated code needed to make any new project activatable for the OnLaunched contract. An app author can opt in to other contracts and activation points as desired via the tool. There are a set of strongly typed Activation methods that the app can override or a loosely typed onActivated method which is the “catch-all”. If you override a strongly typed method, you must mark the args.Handled property as true or the onActivated method will still fire. All activations will always fire the onActivated method (unless one of the strongly methods are marked handled).

4.1.2.5 REFERENCES

1. WinRT UI Activation Spec

4.2 TILES

Tiles are the representation of an application in the Start Menu and the primary way that users launch their applications. Tiles can have the following functionality and behavior:

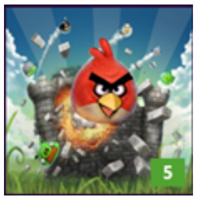
1. Can display new, relevant, and tailored application content to the user through notifications, which allows the user to see at a glance what’s new in their world.
2. Play an integral role in the Start Menu to make Windows feel alive and connected
3. Bring new information that users care about to the top level so they see what’s new in their applications without going to each application to seek for it.
4. Are not only how modern apps represent themselves, but also present themselves to users in a way that is engaging and personal to users and incent them to launch the app.

4.2.1 TILE SIZES AND ANATOMY

There are two tile sizes: square(1x1) and wide(2x1).

4.2.1.1 SQUARE TILE

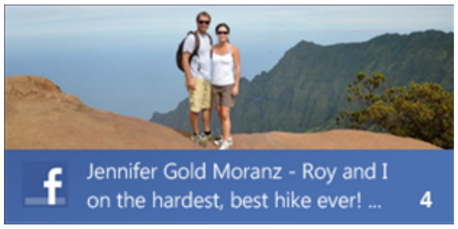
The square tile is the smaller of the two tile sizes and it can contain application branding (either an application icon or name) – as well as full bleed images and potential notification badges (discussed below). Because a square tile contains only basic information, only a limited number of templates are available for updating them.



Square tile size

4.2.1.2 WIDE TILE

This tile size can contain any of the content of a square tile plus richer and more detailed content as well. A broad choice of layout templates is available at this size, to allow the developers to present the rich notifications to the user. If the application wants to use the wide tile, then they must also provide the corresponding square tile as the user can choose to shrink the tile down to the square size to personalize their Start Menu.


Jennifer Gold Moranz - Roy and I
on the hardest, best hike ever! ...
4

Wide tile size

4.2.2 DEFAULT TILES

Default Tiles are the static tiles that included with the application package. If an application doesn't send notifications to their tile, then the default tile is the only tile that is displayed to the user. Even if an application provides a live tile, default tiles are the first thing to be displayed to the user before the tile starts receiving live notifications. It is also used as a fallback for when there are no notifications to be displayed on the tile in cases where the user is offline or other notifications have expired.

Tile UI:

Default tiles are essentially a full bleed image the size of the tile representing the brand/logo of the application. If the application wants to use a wide tile, then the developer must also provide a square tile.

Tile Rendering:

Default tiles are rendered on top of the application color, so if there is any transparency in the default tile image, the application background shows through. The developer can choose to turn the application name on/off for the default tile.

4.2.2.1 CREATING A DEFAULT TILE

The default tile is specified in the installation manifest and the manifest editor in Visual Studio should be used to create the default tile.

4.2.3 TILE NOTIFICATIONS

To create live tiles, developers must choose from a set of pre-defined layouts from the template catalog. Using templates allows tiles to deliver personalized content to the user easily by simply data binding to template fields.

There are two types of templates in the catalog: Static and peek templates

1. **Static templates** have a single frame i.e. one tile's worth of content and so the entire static template is displayed on the screen at the same time.
2. **Peek** templates have multiple frames i.e. more than a tile's worth of content and animate between the different frames at Windows defined intervals. Peek templates can start animating from any of the frames.

Tile notifications specify a template and the images and text which should be inserted into the template. Only Windows provided templates can be used for tile notifications. Users are able to resize tiles to their liking between square and wide sizes. Applications cannot control what size their tile is. Both square and wide tiles can be updated with tile notifications, but the templates are different. **Applications that want to use wide templates MUST include a wide logo in the installation manifest. If an application uses a wide logo, it is STRONGLY RECOMMENDED to send wide templates in notifications. Applications that send tile notifications with wide templates are STRONGLY RECOMMENDED to also include a template for the square size in the same payload since users can resize the tile at any time.** This is accomplished by simply specifying two binding elements in the notification, one with a square template and one with a wide template. A tile that is been resized by the user from the wide size to the square size which receives a notification without a square template specified will instead show the default square tile specified in the manifest.

Logos and App Names in Tile Notifications:

Applications can specify if their logo or application name should be displayed on a per notification basis. *Note – the logo is specified in the installation manifest*. It is recommended that applications choose one and stick with it for visual consistency and so users can easily identify applications. Templates will default to showing the application logo unless specified otherwise in the Tile Notification.

Sending Tile Notifications:

Applications can update their tiles with notifications in one of two ways: using the local WinRT APIs when their application is running or using Push Notifications to send them from their web server.

Cycling Tile Notifications:

Applications can opt-in for Windows to cycle notifications on the tile. Windows will save up to five notifications if this setting is enabled and automatically cycle between them. Notifications will be saved in a FIFO queue according to arrival time. The display order and time for notifications is random and cannot be controlled by applications. Windows uses an algorithm that factors in a variety of variables including number of tiles that have notifications to cycle and how the user is interacting with Windows. Unseen notifications are preferred over seen notifications, and a new notification is always displayed immediately. If a batch of new notifications is delivered to the user, the most recent will be shown immediately, and the other new ones will automatically cycle.

Offline users: If cycling notifications is enabled, up to five notifications will be saved for offline users and delivered when the user comes online. These undelivered notifications also conform to the FIFO queue and replacement logic.

Replacing Tile Notifications:

Notifications can have replacement identifiers which dictate the replacement policy of the saved notifications. These are application specified strings that can be set on both local and push notifications. Windows will examine the tag on a new notification and replace any saved notification with the same tag. Otherwise if there isn't a saved notification with the same tag, the notifications will be replaced in FIFO order.

Clearing Tile Notifications:

Tile notifications can be cleared by a running application. The default tile will be displayed after notifications are cleared. If notification cycling is enabled, all queued notifications will be cleared.

4.2.3.1 TILE NOTIFICATION XML DESCRIPTION

Raising a Tile Notification is remarkably similar to a toast– a developer creates an xml payload based on a well-known template and passes that to a manager object to display or uses push notifications to send the XML payload.

The raw XML for a tile notification that updates the tile with the specified template and content:

```
<?xml version="1.0" encoding="utf-16"?>
<tile>
<visual lang="en-US">
    <binding template="TileWideImageAndText">
        <image id="1" src="package://images\\tile-sdk.png"
alt="package://images/someImage.png"/>
        <text id="1">This tile notification uses local images</text>
    </binding>
    <binding template="TileSquareImage">
        <image id="1" src="package://images\\tile-sdk.png"
alt="package://images/someImage.png"/>
    </binding>
</visual>
</tile>
```

4.2.3.2 <TILE>

Tile notifications are encapsulated with a root element of `tile.` Inside the root element, notifications must specify a `visual` element. Inside the `visual` element, notifications can specify a square template and/or a wide template, denoted by the `binding` element.

4.2.3.2.1 <VISUAL>

The <visual> tag contains all the visual properties of the tile notification including the template to use, custom text, custom images and more.

Visual Attributes

Name	**Description**
version (optional)	(Example: “1”)
lang (optional)	This attribute declares the language of the text in this notification. Windows uses this information to choose the correct font to render the text in. The value must conform to a BCP-47 tag. If omitted, Windows will instead use the Application specific language setting specified in the manifest. (Example: “en-us”)
baseUri	The baseURI can be used to help save space in the your notification payload for specifying

(optional)	images. (Example: http://www.contoso.com/users/871293/notify/89FD3452321D34A/)
branding (optional)	This attribute sets, at a per notification level, the branding to be shown on the tile. (Example: “none”) Options: **none** – nothing is shown **name** – the short name specified in the manifest is displayed **logo** (default value) – the logo specified.

Should I include the language attribute when I send notifications:

- If you are writing a non-localized app, **Omit** the lang attribute when sending notifications. Windows will use the ASLS language since all apps must specify at least one language
- When writing a localized app:
  - Sending localized text in the notification:
    - If this is a cloud notification, **Include** the lang attribute for the language of the text. This will require the app to tell the backend what language this user is running the app in when it registers for channels
    - If this is a local notification, **Omit** the lang attribute. You have already localized the text (hopefully using ASLS), so this should just work
  - Sending MRT resources in the notification:
    - **Omit** the lang attribute, Windows will use the ASLS language, which is what MRT will use to pull the localized text string

4.2.3.2.2 BINDING

A wide and square template binding should be included in one notification. This ensures that the user will always see the notification if they have the tile sized square or wide.

Binding Attributes

Name	**Description**
template (required)	This attribute declares which template should be used. Implicitly, this defines the number of valid text and images elements. (Example: “tile.wide.image”)
lang (optional)	This attribute declares the language of the text in this notification. Windows uses this information to choose the correct font to render the text in. The value must conform to a BCP-47 tag. If omitted Windows will instead use the value specified in the visual lang attribute. (Example: “en-us”)
fallback (optional)	This attributed declares which template should be used in the case that the template named under the *template* attribute is not available on this system in the case that Microsoft has released new templates that the end user’s PC has not downloaded. If omitted, the notification will be dropped if the end user’s PC does not have the template specified in the template attribute. (Example: “tile.wide.image”)

baseUri (optional)	The baseURI can be used to help save space in the your notification payload for specifying images. (Example: http://www.contoso.com/users/871293/notify/89FD3452321D34A/)
branding (optional)	This attribute sets, at a per notification level, the branding to be shown on the tile. (Example: “none”) Options: **none** – nothing is shown **name** – the short name specified in the manifest is displayed **logo** (default value) – the logo specified.

Binding elements are the way that a developer binds custom image and text to their notification. From the example, above:

```
<binding template="TileWideImageAndText">
   <image id="1" src="package://images\\tile-sdk.png" alt="alt text"/>
   <text id="1">This tile notification uses local images</text>
</binding>
<binding template="TileSquareImage">
  <image id="1" src="package://images\\tile-sdk.png" alt="alt text"/>
</binding>
```

Binding Elements	
Text Attributes	**Description**
id (required)	This attribute declares which text field the text should be displayed in.
lang (optional)	This attribute declares the language of the text in this notification. Windows uses this information to choose the correct font to render the text in. The value must conform to a BCP-47 tag. If omitted Windows will instead use the value specified in the binding lang attribute. (Example: “en-us”)
Image Attributes	**Description**
id (required)	This attribute declares which region the image should be displayed in.
src (required)	The path of the image. Images can be specified as JPG (.jpg or .jpeg) or PNG (.png). The baseURI can be used in the visual element to help save space in the your notification payload for specifying images.
alt (optional)	It is recommended that developers set this to specify the alternate text for the image for accessibility purposes.
Image Protocols	**Description**
Package	An image resource included in the package (.appx) file. (Example:

	package://images//welcome.png).
AppData	An image resource in the applications AppData directory. (Example: appdata://images//welcome.png).
http/https	An image resource from the cloud. (Example: http://www.mywebsite.com//images/welcome.png).

4.2.4 SAMPLE JAVASCRIPT CODE

Note: There are currently some known issues where tile updates aren't being painted on the tiles in the PDC-4 builds.

Sending a Tile Notification locally:

```
  var Notifications = Windows.UI.Immersive.Notifications;

  //get a default version of the template
        var tileXml =
Notifications.TileUpdateManager.getTemplateContent(Notifications.TileTempl
ateType.tileWideImageAndText);

  // get the text elements and fill them in
  var tileTextAttributes = tileXml.getElementsByTagName("text");
        tileTextAttributes[0].appendChild(tileXml.createTextNode("This
tile notification uses local images"));

        /*
        You will need to look at the template documentation to know how
many images a particular template has
            - MSDN link
            - you can also use the Advanced Tile SDK Sample to preview all
of the templates
        */

        // get the image elements and fill them in
        var tileImageAttributes = tileXml.getElementsByTagName("image");
        tileImageAttributes[0].setAttribute("src",
"package://images\\tile-sdk.png");

        /*
        Users can resize tiles to square or wide
        Apps can choose to include only square assets (meaning the app's
tile can never be wide), or
        include both wide and square assets (meaning the user can resize
the app's tile to square or wide).
        App's cannot include only wide assets.

        App's that support being wide should include square tile
notifications along with wide notifications since
        users determine the size of the tile.
        */
        // get a filled in version of the square template by using
getTemplateContent
        var squareTileXml =
Notifications.TileUpdateManager.getTemplateContent(Notifications.TileTempl
ateType.tileSquareImage);
        var squareTileImageAttributes =
squareTileXml.getElementsByTagName("image");
        squareTileImageAttributes[0].setAttribute("src",
"package://images\\tile-sdk.png");

        // include the square template into the notification
        var node =
tileXml.importNode(squareTileXml.getElementsByTagName("binding").item(0),
true);
        tileXml.getElementsByTagName("visual").item(0).appendChild(node);

        //create the notification from the XML
        var tileNotification = new
Notifications.TileNotification(tileXml);

        //send the notification to the app's default tile

Notifications.TileUpdateManager.createTileUpdaterForApplication().update(t
ileNotification);
```

Specifying branding in your Tile Notification:

```
  var Notifications = Windows.UI.Immersive.Notifications;
  var tileXml =
Notifications.TileUpdateManager.getTemplateContent(Notifications.TileTempl
ateType.tileWideImageAndText);

  //...
  // set the branding
  var binding = tileXml.getElementsByTagName("binding");
  binding[0].setAttribute("branding", "logo"); // - default, uses the logo
specified in the manifest
  binding[0].setAttribute("branding", "name"); // - use the short name
specified in the manifest
  binding[0].setAttribute("branding", "none"); // - no logo or name

  //...

  var tileNotification = new Notifications.TileNotification(tileXml);

Notifications.TileUpdateManager.createTileUpdaterForApplication().update(t
ileNotification);
```

Having Notifications Cycled by Windows:

```
Windows.UI.Immersive.Notifications.TileUpdateManager.createTileUpdaterForA
pplication().enableNotificationQueue(true);
```

Replacing a Notification:

```
// create tileXml using previous examples
var tileNotification = new Notifications.TileNotification(tileXml);

// set the tag of the notification
tileNotification.tag = "msft";


Notifications.TileUpdateManager.createTileUpdaterForApplication().update(t
ileNotification);
```

Clearing Tile Notifications:

```
Windows.UI.Immersive.Notifications.TileUpdateManager.createTileUpdaterForA
pplication().clear();
```

4.2.5 REFERENCES

1. SDK samples
   a. JavaScript – \\windesign\modern\tdonahue\SDK samples
2. M3 tile templates
3. wpndisc

4.3 SECONDARY TILES

Secondary Tiles enable users to promote interesting content and deep links from Immersive Applications onto the Modern Start Menu. A deep link is a reference that can be used to navigate back to a specific location inside of the pinning application. Secondary Tiles are always associated with their parent application – they may never be “re-parented” to another application. Pinning Secondary Tiles to Modern Start Menu provides users with a consistent and efficient way to launch directly into a frequently used area of their favorite application experiences—either a general sub section of the parent application containing content which is updated frequently or a deep link to some very specific area of the application the user is especially interested in keeping track of.

Canonical examples of secondary tile usage include:

- A weather application exposing weather updates for a specific city or region
- A calendar application exposing upcoming events or meetings
- A social application surfacing status and updates of important contacts

- An RSS reader allowing pinning of specific feeds or user defined categories

Secondary Tiles enable users to personalize their Modern Start Menu experience with playlists, photo albums, buddies, and other items they find important.

More generally, any content exposed by an immersive application that will change frequently and which a user will want to keep an eye on as changes occur is a good candidate for exposing as a potential Secondary Tile. Once pinned, users may get at-a-glance updates on their tiles and directly launch into their application to reveal a focused experience centered on the pinned content or contact.

Users always have explicit control of Secondary Tile creation; applications cannot create Secondary Tiles programmatically. Users also have explicit control over Secondary Tile removal - either through the Modern Start Menu or within the parent application.

It is easy for developers to leverage the app bar as a surface to pin and unpin content to the Modern Start Menu. Windows supplies a suggested implementation of the necessary javascript and a button to match the personality of the app bar. If there is a compelling scenario for not using the app bar, the projected WinRT function may be called via javascript from app surfaces outside the App Bar.

1. Enabling or disabling the pin button, whether on the app bar or another surface within the application is left to the application. The following guidance should be adhered to when choosing to enable the pin button: If the content in focus is pinnable, the app bar should show and enable the pin button.
2. If the content in focus is already pinned, the app bar should show and enable the pin button so that the user can use it to unipin the pinned content.
3. If the content in focus is not pinnable, the app bar Pin button should not be shown. If the application exposes the pin command outside the app bar, the app bar's Pin button should either not be shown, or be shown in a disabled state, depending on the UI surface and scenario where it will appear when it becomes enabled.

Open Issue: As of PDC-4, the app bar is still working to get inbox icons and tasks implemented. Expect the app bar functionality and corresponding icons for pin/remove to be added at a subsequent date.

Secondary Tiles, like all tiles on the Modern Start Menu, are dynamic outlets that should be frequently updated with new content. Secondary Tiles may surface notifications and updates using the same mechanisms as any other tile. To update the tile when the application is not running, the Secondary Tile must request and open a channel URI with the Microsoft Notification service.

Secondary Tiles should not be used as shortcuts to discrete files that cannot change.

Pinning is a contract between a MoSh application and the Windows pinning infrastructure. The following is a high-level overview of the pin flow**:**

1. The user invokes pin command through the application, which in turn calls the WinRT ContentTileManager API's pin method.
2. The application provides the information necessary to create a Secondary Tile.
3. The OS shows a fly-out depicting a preview of the tile and asking the user to confirm the creation of the Secondary Tile.
4. User clicks "yes" and the Pin Infrastructure creates the Secondary Tile in the Modern Start Menu.
5. The application opens a channel URI for receiving notifications from the Secondary Tile.

The unpin flow is similar**:**

1. The user invokes the unpin command through the application, which in turn calls the WinRT pin API.
2. The application provides the necessary information to remove a Secondary Tile.
3. The OS shows a fly-out with preview of the tile to be removed and asks the user to confirm
4. The user clicks "yes" and the pin infrastructure removes the Secondary Tile from the Modern Start Menu.

The two key points to note are that it is up to the application to provide the pin infrastructure with the right information to create or remove a Secondary Tile, and the user is in control of pinning the tile or dismissing the tile pin fly-out.

The application should use meaningful, recreatable IDs for secondary tiles. This is important for the following reasons:

1. Secondary Tiles may be re-acquired by users when the application is installed on a second PC. Using predictable Secondary Tile IDs that are meaningful to an application will help the app understand what to do with these tiles when they are seen within a fresh installation on a new PC.
2. At runtime, the application may query if a specific tile exists. By using meaningful IDs for Secondary tiles, the application may properly ask, "Hey, is this a tile that the user has already pinned?"
3. The Secondary Tile platform may be asked to return the set of all Secondary Tiles belonging to an application. Using meaningful IDs for these tiles will help the application reason over the tiles and perform appropriate actions.

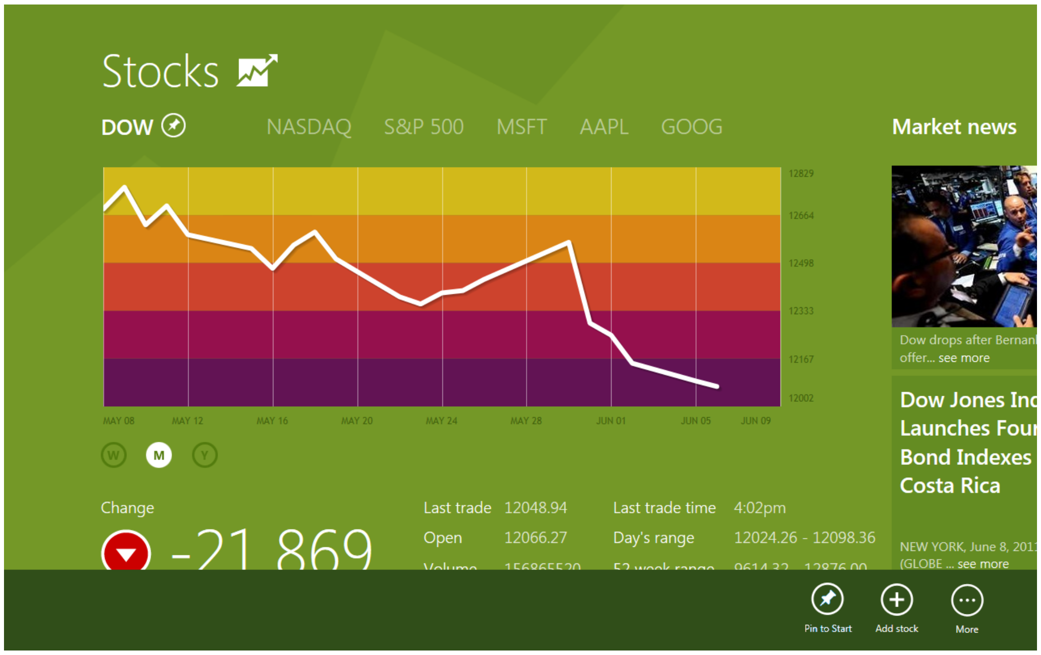
As described in the preceding section, Secondary Tiles are similar to tiles, but there are some differences. The following list describes some of those similarities:

- Same elements and capabilities as any other tile
  - Has a Logo and Small Logo.
  - May show notifications and badges.

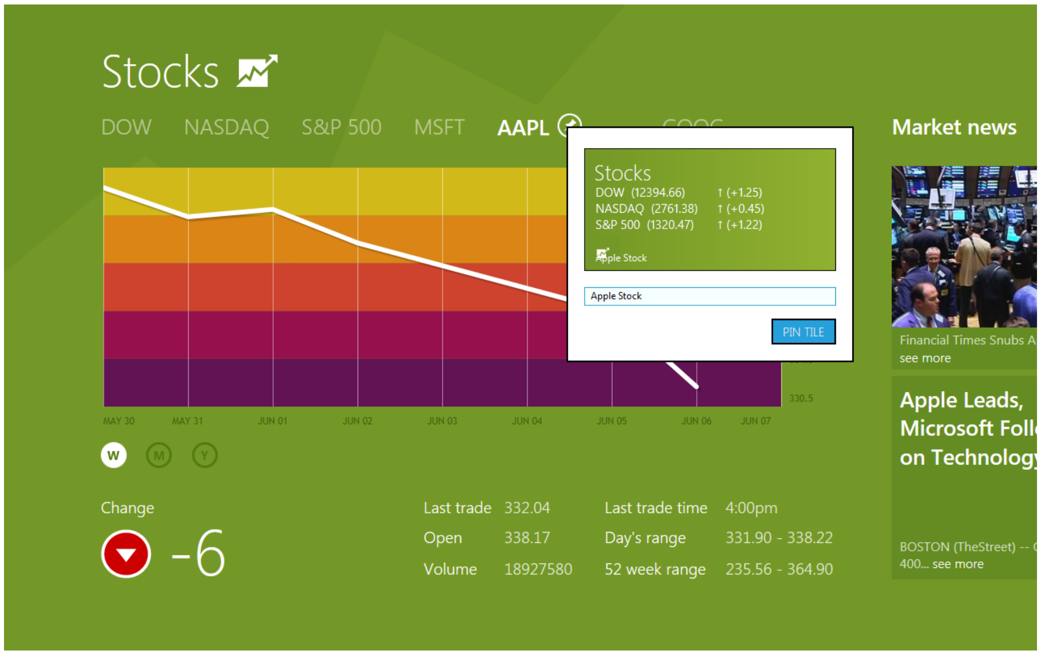
- Uses templates to determine layout.
- Must include a 152x152 Logo
- May optionally use a 312x152 LogoSecondary Tiles may be re-arranged on the Start Menu
- Secondary Tiles may be moved to the All Programs View or Start
- Secondary Tiles will automatically be deleted when the app is uninstalled.

Secondary Tiles are different from Application Tiles in some noticeable ways:

- Users may delete their Secondary Tiles at any point in time without deleting the parent application
- The application may initiate the addition of a Secondary Tile at runtime.
  - Regular application Tiles are only created at install time.
  - The following image depicts an application bar to which a developer could add a pin button. For more information, please refer to App Bar section of this document.



  - A default glyph for pin and remove will be supplied for developers.
  - Developers may also add Secondary Tiles through contextual interactions specific to their app.
    - This is also reflected in the UI above.
- A flyout always prompts the user for confirmation when adding a Secondary Tile which is different from a regular application tile.
  - See below for an example of this flyout in the context of an application



Last trade 332.04 Last trade time 4:00pm
Open 338.17 Day's range 331.90 - 338.22
Volume 18927580 52 week range 235.56 - 364.90

4.3.1 OVERVIEW OF THE SECONDARY TILES CONTRACT—SOURCE APP

The source application handles activation requests from Modern Start Menu for the Secondary Tile. This is accomplished via the standard activation mechanism used for tiles. For more information on activation refer to the Modern Activation section. Secondary Tiles may not be activated through any mechanism except the Modern Start Menu.

The source app is also responsible for:

- Creating an ID for referring to the Secondary Tile.
- Requesting a Channel URI in order to serve notifications to the Secondary Tile.
- Exposing the pin/unpin command appropriately.
- Handling activation requests for the deep links that are exposed by the Secondary Tile.

4.3.2 SAMPLE JAVASCRIPT CODE

4.3.2.1 CREATING A SECONDARY 1X1 TILE

This code demonstrates how to create a 1x1 tile with a different foreground and background color that the parent application tile.

```
// Create a 1x1 secondary tile
var logoUri = new Windows.Foundation.Uri("ms-resource://logo.png");
var tile = new Windows.UI.Immersive.SecondaryTile(
                "Microsoft.Sample.StockApp.msft",
                "MSFT",
                "Stocks - Microsoft",
                "stock=msft,view=detailed",
                Windows.UI.Immersive.TileDisplayAttributes.showName,
                logoUri);

tile.requestCreateAsync().then(
    function() {
        updatePageStatus("Tile created.");
    });
```

4.3.2.2 CREATING A SECONDARY WIDE TILE

This code demonstrates how to create a wide tile that will be registered to receive notifications.

```
// Create a wide secondary tile
var logoUri = new Windows.Foundation.Uri("ms-resource://logo.png");
var wideLogoUri = new Windows.Foundation.Uri("ms-
resource://widelogo.png");
var tile = new Windows.UI.Immersive.SecondaryTile(
                "Microsoft.Sample.StockApp.msft",
                "MSFT",
                "Stocks - Microsoft",
                "stock=msft,view=detailed",

Windows.UI.Immersive.TileDisplayAttributes.dynamicTileCapabe,
                logoUri,
                wideLogoUri);

tile.requestCreateAsync().then(
    function() {
        updatePageStatus("Secondary tile created!");
    });
```

4.3.2.3 RETRIEVING A LIST OF SECONDARY TILES

This code demonstrates how to retrieve a list of secondary tiles for an application in the package.

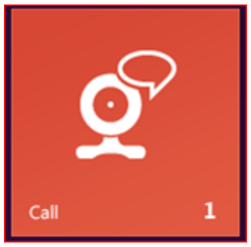
```
// Get secondary tile ids for an application
Windows.UI.Immersive.SecondaryTile.findAllAsync().then(
    function(tiles) {
        tiles.forEach(function(tile) {
            doSomething(tile);
        });
    });
```

4.3.3 REFERENCES

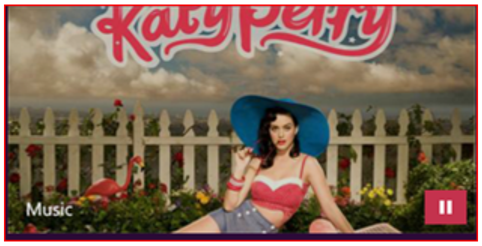
1. Secondary Tiles API/DDI Doc
2. Secondary Tiles Functional Spec
3. Secondary Tiles Dev Design Spec

4.4 TILE BADGES

Badges show summary information such as the number of new items available to the user in a specific application context (E.g. Mail app can show 9 to indicate to the user that they have 9 unread emails). Badges can be displayed on both tile sizes. They can be numeric (0-99) or one of a set of Windows-provided glyphs e.g. the pause glyph indicates to the user that their music is currently paused in the music app.



Numeric badge



Glyph Badge

4.4.1 BADGE NOTIFICATIONS

Badge notifications specify either a number or Windows provided glyph. This appears as a badge overlay on the application's tile.

Displaying a Count:

The raw XML for a badge notification that updates the badge overlay on a tile with the number 20

```
<?xml version="1.0" encoding="utf-8"?>
<badge value="20"/>
```

Valid numbers for using a count are (0, 99). Numbers greater than 99 will appear as *99*

Displaying a Glyph:

The raw XML for a badge notification that updates the badge overlay on a tile with a glyph

```
<?xml version="1.0" encoding="utf-8"?>
<badge value="available"/>
```

Valid glyphs are

1. None
2. Activity
3. Available
4. Away
5. Busy
6. NewMessage
7. Paused
8. Playing
9. Unavailable
10. Error

Clearing Badge Notifications:

Badge notifications can be cleared by a running application using local WinRT APIs. Note: sending a badge notification with the value of “0” or “none” will also clear the badge.

4.4.2 SAMPLE JAVASCRIPT CODE

Note: There are currently some known issues where tile badge updates aren’t being painted on the tiles in the PDC-4 builds.

Sending a badge with a number:

```
var Notifications = Windows.UI.Immersive.Notifications;
var badgeXml =
Notifications.BadgeUpdateManager.getTemplateContent(Notifications.BadgeTem
plateType.badgeNumber);
var badgeAttributes = badgeXml.getElementsByTagName("badge");
badgeAttributes[0].setAttribute("value", "6");
var badge = new Notifications.BadgeNotification(badgeXml);

Notifications.BadgeUpdateManager.createBadgeUpdaterForApplication().update
(badge);
```

Sending a badge with a glyph:

```
var Notifications = Windows.UI.Immersive.Notifications;
var badgeXml =
Notifications.BadgeUpdateManager.getTemplateContent(Notifications.BadgeTem
plateType.badgeGlyph);
var badgeAttributes = badgeXml.getElementsByTagName("badge");
badgeAttributes[0].setAttribute("value", "available");
var badge = new Notifications.BadgeNotification(badgeXml);

Notifications.BadgeUpdateManager.createBadgeUpdaterForApplication().update
(badge);
```

Clearing a badge notification:

```
Windows.UI.Immersive.Notifications.BadgeUpdateManager.createBadgeUpdaterFo
rApplication().clear();
```

4.4.3 REFERENCES

1. M3 tile templates
2. wpndisc

4.5 LOCK SCREEN NOTIFICATIONS

The lock screen can also display tile and badge notifications an application sends. For this functionality to be enabled, users must manually allow apps to display tile or badge notifications on the lock screen in the Lock Screen tab of the Personalization page in Control Panel.

Lock Screen Tile Notifications:

Only the text in the notification will be displayed; the images will not be displayed. Applications must opt into lock screen notifications in the manifest by choosing either TileText or BadgeAndTileText in the visual elements.

Lock Screen Badge Notifications:

The lock screen can also display badge notifications. A monochrome badge logo must also be supplied (this is a Store certification requirement). Like Lock Screen Tile Notifications, applications must opt into lock screen notifications in the manifest by choosing either Badge or BadgeAndTileText in the visual elements.

4.5.1 SAMPLE CODE

Lock Screen Manifest Markup for displaying Tile Text:

```
<VisualElements …>
   <LockScreen Notification="tileText"/>
   <SplashScreen Image="images\splashscreen.png" />
</VisualElements>
```

Lock Screen Manifest Markup for displaying a Badge:

```
<VisualElements …>
   <LockScreen BadgeLogo="images\badgeLogo.png" Notification="badge"/>
   <SplashScreen Image="images\splashscreen.png" />
```

```
</VisualElements>
```

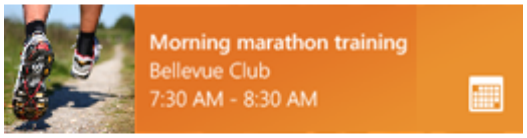
Lock Screen Manifest Markup for displaying a Badge and Tile Text:

```
<VisualElements …>
   <LockScreen BadgeLogo="images\badgeLogo.png"
Notification="badgeAndTileText"/>
   <SplashScreen Image="images\splashscreen.png" />
</VisualElements>
```

4.5.2 REFERENCES

1. M3 tile templates
2. wpndisc

4.6 TOASTS



Toast notifications are transient notifications that interrupt the user with relevant, time-sensitive information and provide a quick way to directly access that content in an application. Toast is great for Incoming email alerts, IM requests and breaking news. When you think about how you'll use a toast notification, consider that the user may not see it.

Toasts provide developers with the capability to use their application to raise a notification. This involves two steps:

1. Set the **ToastCapable** attribute of Visual Elements in your app's manifest to true
2. Create the toast object using Xml markup
3. Supply the toast notification via Local WinRT APIs detailed below or using Push Notifications (Section X.X).

That's it. These steps create a new toast on the screen.

4.6.1 TOAST XML DESCRIPTION

Raising a Toast Notification is remarkably similar to a tile update – a developer creates an xml payload based on a well-known template and passes that to a manager object to display. Toast is visually distinct from a tile but the markup is similar. The most basic mark up for toast looks like this:

```
<toast launch="/update.html?id=89FD3452321D34A">
```

```
       <visual
baseUri="http://www.contoso.com/users/871293/notify/89FD3452321D34A/"
            version="1">
           <binding template="Toast.Picture.01">
                 <image id="1"
                        src="package://runner.png"
                        alt="This is an image" />
                 <text id="1">Morning Marathon Training</text>
                 <text id="2">Bellevue Club</text>
                 <text id="3">7:30 AM - 8:30 AM</text>
           </binding>
       </visual>
</toast>
```

4.6.1.1 <TOAST>

The <toast> tag is what defines this as a toast. Inside the tag, a developer can include a launch context property. This is a string that is passed to your application if the user activates (clicks on) the toast. The format is:

```
Launch="custom string"
```

This string is only for third party apps and has no meaning to Windows.

4.6.1.1.1 <VISUAL>

The <visual> tag contains all the visual properties of the toast notification including the template to use, custom text, custom images and more.

4.6.1.1.1.1 BINDING

The template attribute is how the developer declares what UI template will be used by the toast.

Binding Attributes

Name	**Description**
template (required)	This attribute declares which template should be used. Implicitly, this defines the number of valid text and image elements.

Binding elements are the way that a developer binds custom image and text to their notification. From the example, above:

```
<binding template="Toast.Picture.01">
  <image id="1" src="package://runner.png" alt="This is an image" />
  <text id="1">Morning Marathon Training</text>
  <text id="2">Bellevue Club</text>
  <text id="3">7:30 AM - 8:30 AM</text>
```

```
</binding>
```

Binding Elements

Text Attributes	**Description**
id (required)	This attribute declares which text field the text should be displayed in.
Image Attributes	**Description**
id (required)	This attribute declares which region the image should be displayed in.
src (required)	The path of the image. Images can be specified as JPG (.jpg or .jpeg) or PNG (.png). The baseURI can be used in the visual element to help save space in the your notification payload for specifying images.
alt (optional)	It is recommended that developers set this to specify the alternate text for the image for accessibility purposes.
Image Protocols	**Description**
Package	An image resource included in the package (.appx) file. (Example: package://images//welcome.png).
AppData	An image resource in the applications AppData directory. (Example: appdata://images//welcome.png).
http/https	An image resource from the cloud. (Example: http://www.mywebsite.com//images/welcome.png).

4.6.1.1.2 <AUDIO>

Since Toasts have the ability to play sound, the markup will also describe the audio that should accompany the visual. Note: if no audio option (or tag) is specified, the system will play the default sound.

Audio Attributes

Name	**Description**
src (optional)	Media file to play (system files only). (Example: src="ms://boing.wav").
exit (optional)	Ringing toasts only. Media file to play when Toast is dismissed. (Example: exit="ms://whoosh.wav").
loop (optional)	Ringing toasts only. Boolean to denote if playing the media file should repeat. False by default. (Example: loop="true").
silent (optional)	Used to indicate that this toast should play no sound. Overrides

	src tag. (Example: silent="true").

4.6.2 WINRT API DETAILS

ToastNotificationManager Object

A factory class that gives you default content and produces ToastNotifiers for a given application

Method	**Description**
GetDefaultContent	Get a DOM containing the default XML for the specified template.
CreateToastNotifierForApplication	Create a ToastNotifier object tied to the current application.

ToastNotifier Object

A class that can raise and hide Toast for a given application as well as managing Toast Notification settings for that app.

Method	**Description**
Setting	
Show	Schedules the toast for display – note that the OS may delay the actual rendering of the toast based on other system activity.
Hide	Remove the toast from the screen. Used primarily for communication applications to respond to a call being answered.

ToastNotification Object

An object representing a toast notification. This can be shown using an instance of ToastNotifier.

Property	**Description**
Content	Allows developers to manipulate the XML template that defines the appearance and behavior of the Toast Notification.
ExpirationTime	Defines how long the toast is valid for. If the toast has not been shown by expirationTime, it will be discarded. This makes little sense for toast notifications.

4.6.3 EXAMPLE CODE

The following code shows the markup for a Toast notification.

```
<toast launch="/update.html?id=89FD3452321D34A">
       <visual
baseUri="http://www.contoso.com/users/871293/notify/89FD3452321D34A/"
            version="1">
           <binding template="Toast.Picture.01">
                 <image id="1"
```

```
                         src="package://runner.png"
                         alt="This is an image" />
                  <text id="1">Morning Marathon Training</text>
                  <text id="2">Bellevue Club</text>
                  <text id="3">7:30 AM - 8:30 AM</text>
            </binding>
        </visual>
</toast>
```

This shows the required manifest markup in order to support sending Toasts:

```
<VisualElements … ToastCapable="true">
   <SplashScreen Image="images\splashscreen.png" />
</VisualElements>
```

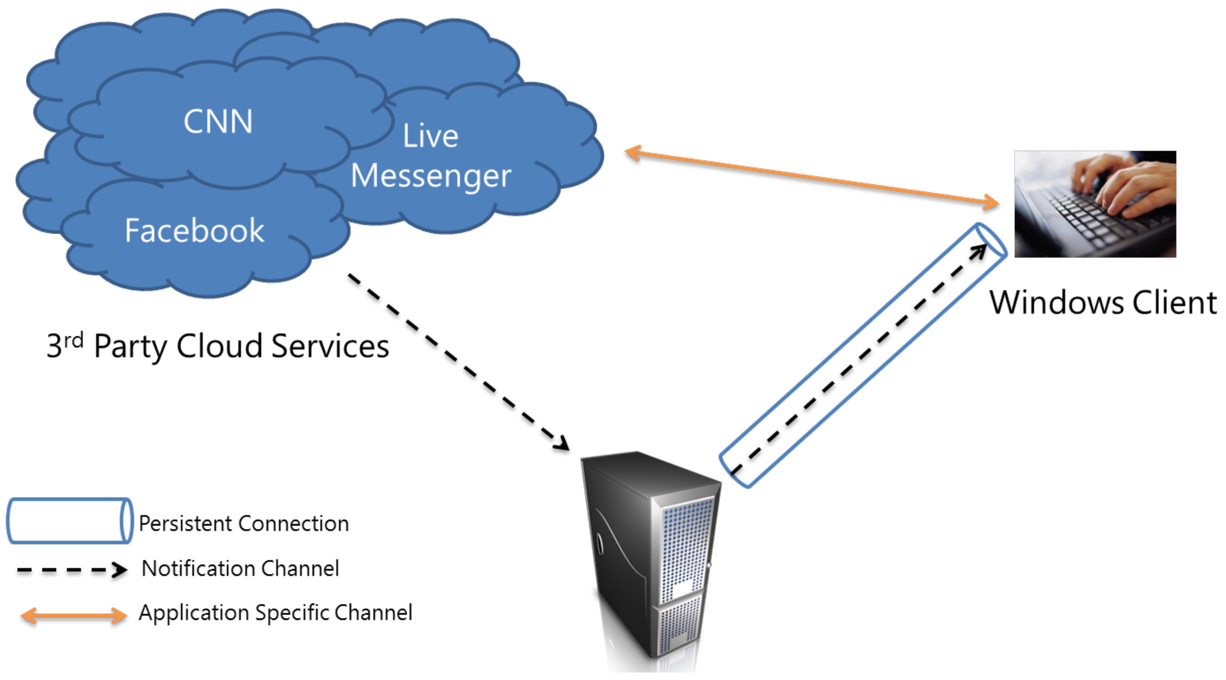
This code shows how to send a toast locally using WinRT APIs.

```
var Notifications = Windows.UI.Immersive.Notifications;
var notificationManager = Notifications.ToastNotificationManager;

var toastXml =
notificationManager.getTemplateContent(Notifications.ToastTemplateType.toa
stSmallImageAndText04);

var strings = toastXml.getElementsByTagName("text");
strings[0].appendChild(toastXml.createTextNode("Morning marathon
training"));
strings[1].appendChild(toastXml.createTextNode("Bellevue Club"));
strings[2].appendChild(toastXml.createTextNode("7:30 AM - 8:30 AM"));

var images = toastXml.getElementsByTagName("image");
images[0].setAttribute("src",
"http://2.bp.blogspot.com/_FkWuK4k5RyI/Sdfzlv_C15I/AAAAAAAAAMQ/cfSSIB9gz-
8/s400/MichiganStateLogo.jpg");

var toast = new Notifications.ToastNotification(toastXml);
notificationManager.createToastNotifier().show(toast);
```

4.6.4 REFERENCES

1. Notifications Concept Document
2. Notifications XML API-DDI
3. Notifications API API-DDI

4.7 PUSH NOTIFICATIONS



Push Notifications can be used to send a toast or tile notification from a webserver without the application having to run on the local machine. To enable push notifications on the client, the developer needs to do three things:

1. Request a channel using WinRT Notifications APIs
2. Send that channel to your application server
3. Use the application server to POST an XML payload to the channel via WNS

4.7.1 REQUESTING A CHANNEL

A notification channel is simply a URI that the application can use to send a notification to an individual user's specific machine. In order for an application to obtain a channel, it must use WinRT APIs. Details on the WinRT APIs for requesting a channel can be found below:

PushNotificationChannelManager Object

Method	**Description**
CreatePushNotificationChannelForApplicationAsync	Request a channel from WNS for this application. This closes any previous channels requested by the application. This Async operation returns a CreatePushNotificationChannelOperation object which can be used to initiate a channel request.

PushNotificationChannel Object

Represents a channel that an app server can use to post messages to a given tile.

Property	Description
Uri	The URI to which a 3rd party app server should push a notification. This is what should be transmitted to and stored by the third-party app server.
ExpirationTime	Defines how long the channel is valid for. At expirationTime the channel will behave as though it has been closed. Since channels are immutable, they cannot be renewed and must be replaced with a call to createPushNotificationChannelForApplicationAsync..
Method	**Description**
Close	Invalidate this channel. Any notifications pushed to this channel will be dropped at the service. To resume sending push notifications to this application, the app must request a new channel.

4.7.2 STORING A CHANNEL

There is good news and bad news in storing your channel. The bad news is that channel storage is very dependent on your application semantics and implementation – we can’t really give useful sample code. The good news is that storing a channel is pretty straightforward and involves remembering two pieces of data – a URL of max length *nnnnn* and (optionally) a date at which that URL becomes invalid.

4.7.3 SENDING A TOAST OR TILE NOTIFICATION FROM THE SERVER

The job of the third-party app server is to store channels and push notifications when relevant events occur. When sending a notification via WNS, the third party app server should set the appropriate headers, and then post the notification XML to the channel URI. Relevant headers are described, below.

4.7.3.1 HTTP POST HEADERS

The sender must set the following headers as well as setting the content type to text/xml.

HTTP POST Headers

Header	Description
X-WNS-Type	Tells the server what type of notification is in the payload. Values: wns/tile, wns/badge, wns/toast, wns/raw
X-WNS-Priority	This tells WNS how to batch notifications when the client is unavailable. For windows 8, the value must be set according to notification type (e.g. no normal priority toast or high-priority badges).

	Values: high – for toast; normal – for all other notifications.
X-WNS-TTL	This is the time (in seconds) for which the toast is valid. It should be non-zero. Values: Positive integers

4.7.3.2 HTTP POST PAYLOAD

For toast, tile, and badge updates, this is the XML template described in the sections covering those areas. For RAW notifications, it is up to you.

4.7.3.3 HTTP POST STATUS CODES AND ERRORS

Once you post a message to the WNS channel URI, you'll get back one of the following responses:

HTTP POST Status Codes

Code	Description
200/success	WNS received your notification and will try and deliver appropriately. Note that this does not necessarily mean that it will arrive at the client.
400/bad request	
401/unauthorized	The secret key that you provided to WNS is not authorized to send to the channel URI to which you posted.
405/method not allowed	You tried to access the URI using GET instead of POST.
500/internal server error	Problem with WNS. Please contact support.

4.7.4 EXAMPLE CODE

The following shows the code for obtaining a channel using WinRT APIs:

```
var PushNotifications = Windows.Networking.PushNotifications;

var channelOperation =
PushNotifications.PushNotificationChannelManager.createPushNotificationCha
nnelForApplicationAsync();

channelOperation.then(function (newChannel) {
      // store channel
      var expiration = newChannel.expirationTime;
      var uri = newChannel.uri;
   },
   function (error) {
      // Error obtaining Channel
   }
);
```

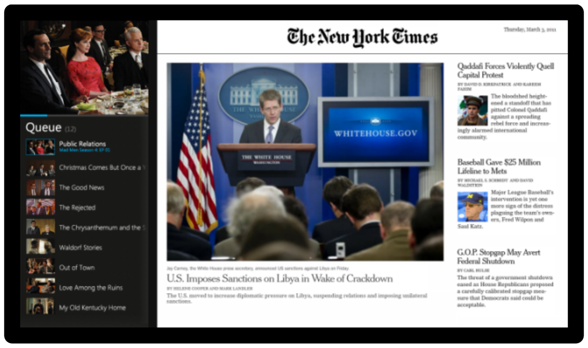
4.7.5 REFERENCES

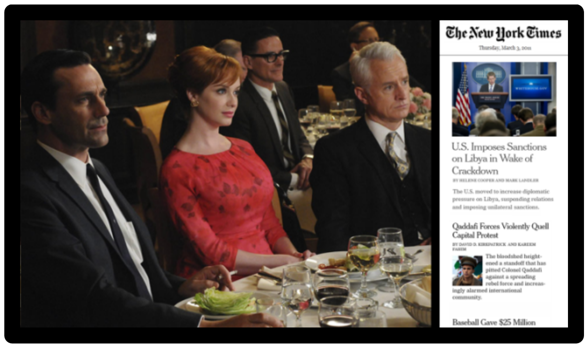
1. Notifications Concept Document
2. Notifications XML API-DDI
3. Notifications API API-DDI

4.8 SNAPPING

In the immersive environment, the user can see either one or two apps on screen at a time. When there are two apps on screen, one of them will be big and the other will be small. The small one is called the "snapped" app. The act of making an app small is called "snapping".

Snapping is what allows the user to multitask with two apps visible at the same time. A snapped app is always a standard width, measured in touch independent pixels (TIPs), and can be placed on either side of the screen. The app that shares the screen with the snapped app fills the remaining display area.


Queue
The New York Times
U.S. Imposes Sanctions on Libya in Wake of Crackdown
Qaddafi Forces Violently Quell Capital Protest
Baseball Gave $25 Million Lifeline to Mets
G.O.P. Stopgap May Avert Federal Shutdown


The New York Times
U.S. Imposes Sanctions on Libya in Wake of Crackdown
Qaddafi Forces Violently Quell Capital Protest
Baseball Gave $25 Million

App snapped left App snapped right

It's trivial for the user to change which app is big and which app is small by manipulating the splitter between the two apps.

4.8.1 APPLICATION LAYOUT

An application can be displayed in one of three layouts:

- **Snapped** – 320 touch independent pixels (TIPs) wide, at the left or right side of the screen. It is always the full height of the screen.
- **Fill** – Fills the portion of the screen not occupied by the snapped app. In a landscape orientation, all fill states are always a minimum of 1024px wide.
- **Full screen** – Only one app visible, spanning edge to edge

A user can put an app into any one of these states *at any time*.

4.8.2 HOW TO THINK OF THE SNAPPED STATE

The snapped app is the app resized. It is not a gadget. All immersive apps can be snapped, just as all windows can be Aero Snapped in Windows 7. However, unlike Aero Snap, the snapped state in the immersive shell is a fixed width, so the app developer can design and target for that specific size. The app developer doesn't have to design for the awkward widths between snapped and fill states.

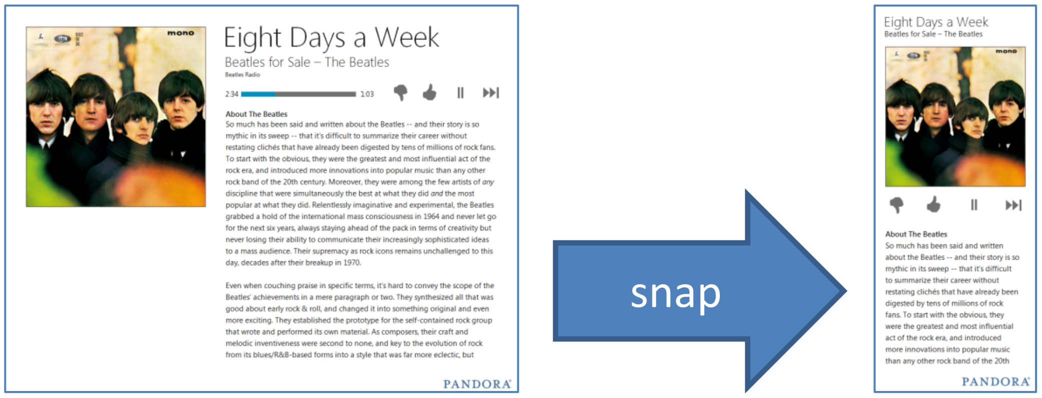
If the app doesn't design for the snapped state, the app's window will still be resized. Content can be cropped and scrollbars may be introduced.

The Windows 8 multitasking value prop depends on applications providing value when they are snapped.

4.8.3 DESIGNING FOR THE SNAPPED STATE

1. **Maintain state and preserve continuity**
   Snapping and unsnapping is trivially easy for the user and they can do it at any time. Your app can be snapped even when it wasn't the user's specific intention to snap you; they might just want to make the other app bigger. Because of this, snapping and unsnapping should never destroy the user's "work".

   All layouts, pages, and views supported by your app in fill and full-screen states should ideally have corresponding snapped versions. Snapped views don't need to be – and often cannot be – identical to the fill and full-screen layouts, but the overall context should be maintained.



Eight Days a Week
Beatles for Sale – The Beatles
Beatles Radio

About The Beatles
So much has been said and written about the Beatles -- and their story is so mythic in its sweep -- that it's difficult to summarize their career without restating clichés that have already been digested by tens of millions of rock fans. To start with the obvious, they were the greatest and most influential act of the rock era, and introduced more innovations into popular music than any other rock band of the 20th century. Moreover, they were among the few artists of *any* discipline that were simultaneously the best at what they did *and* the most popular at what they did. Relentlessly imaginative and experimental, the Beatles grabbed a hold of the international mass consciousness in 1964 and never let go for the next six years, always staying ahead of the pack in terms of creativity but never losing their ability to communicate their increasingly sophisticated ideas to a mass audience. Their supremacy as rock icons remains unchallenged to this day, decades after their breakup in 1970.

Even when couching praise in specific terms, it's hard to convey the scope of the Beatles' achievements in a mere paragraph or two. They synthesized all that was good about early rock & roll, and changed it into something original and even more exciting. They established the prototype for the self-contained rock group that wrote and performed its own material. As composers, their craft and melodic inventiveness were second to none, and key to the evolution of rock from its blues/R&B-based forms into a style that was far more eclectic, but

Eight Days a Week
Beatles for Sale – The Beatles

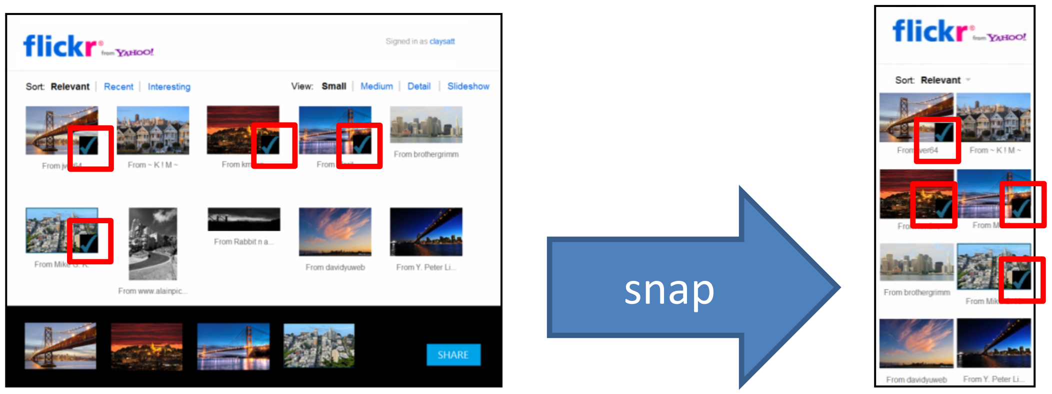
About The Beatles
So much has been said and written about the Beatles -- and their story is so mythic in its sweep -- that it's difficult to summarize their career without restating clichés that have already been digested by tens of millions of rock fans. To start with the obvious, they were the greatest and most influential act of the rock era, and introduced more innovations into popular music than any other rock band of the 20th

   In this example using Pandora, although the snapped state hides the timeline, resizes the image, and reflows the layout, the user maintains overall context.

   <u>But you say:</u>
   - *"I don't have the resources to design <insert obscure feature> for the snapped state."*

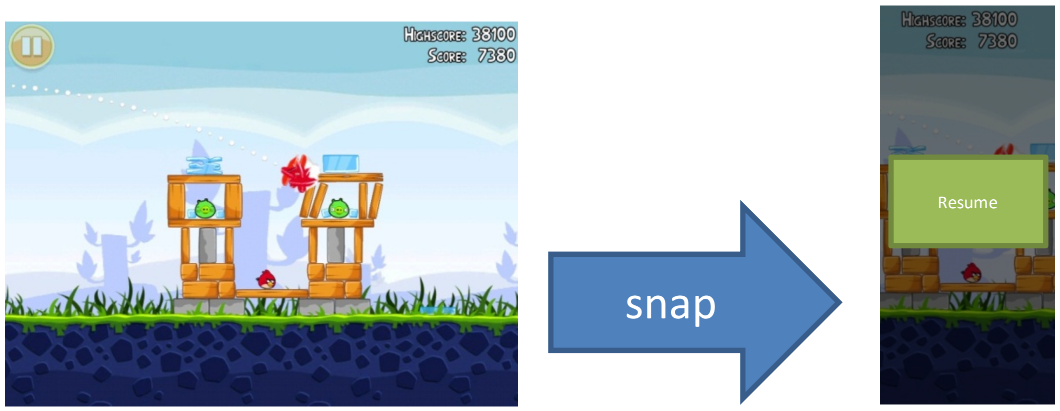
- *"I tried but simply can't make this feature work in this small state!"*

Recommendation:
Maintain the state at least, even if it means showing reduced functionality in the snapped view. If you absolutely cannot build a tailored snapped view for a page, prioritize maintaining state over showing the user a more attractive but otherwise out of context snapped page.


flickr from YAHOO!
Sort: Relevant | Recent | Interesting
View: Small | Medium | Detail | Slideshow
SHARE
snap

What we DO want: In this example using Flickr, even though the list of selected of photos at the bottom of the screen is not available in the snapped state, which forces reduced functionality, the user's selections are preserved.

What we DON'T want: Whenever Flickr is snapped, it sends the user to a beautifully-designed, full-featured snapped view of its main Photostream page, regardless of what the user was seeing before that.

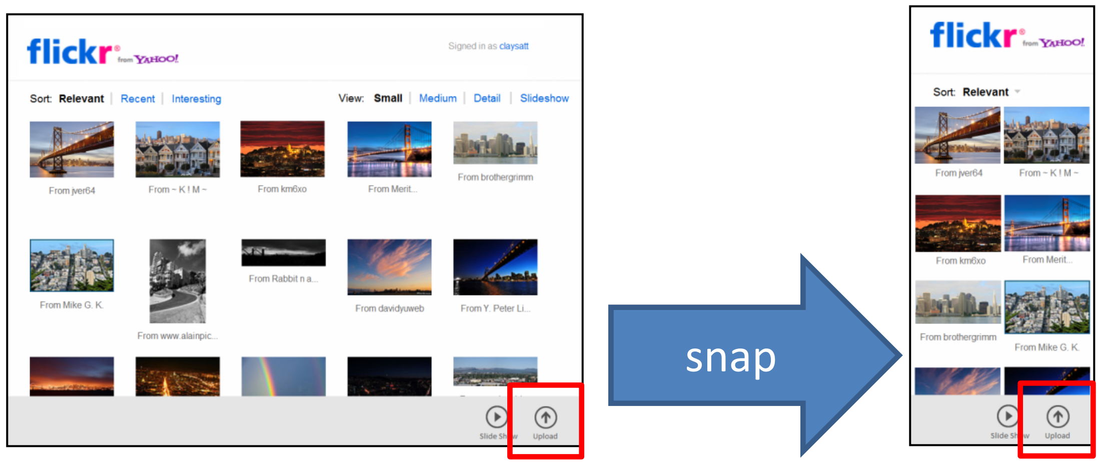


What we DO want: In this example using Angry Birds, when the app is snapped, the user cannot continue playing (reduced functionality), but the state of the game is preserved.

What we DON'T want: Whenever Angry Birds is snapped, it sends the user to a beautifully-designed full-featured snapped state of its Top Scores page.

2. **Have feature parity across states**
Strive to have feature parity across snapped and unsnapped states. Remember that snapping in the immersive environment is a parallel to Aero Snapping in the classic Desktop; the user does not expect to lose features when they snap your app.

In those cases where you simply cannot support a feature in the snapped state, we recommend keeping an entry point to that feature and programmatically unsnapping the app when the user triggers that entry point.


flickr from YAHOO!
Signed in as claysatt
Sort: Relevant | Recent | Interesting
View: Small | Medium | Detail | Slideshow
snap
Sort: Relevant
Slide Show
Upload

In this example using Flickr, the **Upload** button launches the Picker. There is no snapped version of the Picker, but it is better to leave the entry point to the Upload feature and programmatically unsnap when necessary than to omit the feature altogether in the snapped state.

3. **Put the user in control**
The choice of the app's layout state should be left to the user, except in special cases. Reserve the ability to programmatically unsnap your app for situations when it is absolutely necessary to do so. (See below for more details how to programmatically unsnap) For instance, the scenario mentioned above in which Flickr could only provide the Picker when in a nonsnapped state is one in which there is a justification for leaving the snapped state programmatically. If your app has snapped views for all possible pages in the app, there should be no need to use programmatic unsnapping at all.

4. **Use media queries for your layout logic**

Take advantage of the media queries technology[1] to drive your layout logic. See the **View Sizing and Scaling** section for more details on using media queries. The following example show how you can use media queries in applying CSS styles to the different layout states.

```
@media (win-ui-layout: snapped) {
    // CSS styles for the snapped layout state
}

@media (win-ui-layout: filled) {
    // CSS styles for the fill layout state
}

@media (win-ui-layout:fullscreen) {
    // CSS styles for the full screen layout state
}
```

Note: This win-ui-layout media query is not yet implemented. In the meantime, you can use the existing max-width media queries to simulate this.

```
// The following assumes a device with target resolution of 1366x768

@media (max-width: 320px) {
    // CSS styles for the snapped layout state
}

@media (min-width: 321px) and (max-width: 1024px) {
    // CSS styles for the fill layout state
    }

@media (min-width: 1025px) {
    // CSS styles for the full screen layout state
}
```

When do I use media queries and when do I use JavaScript layout change events?

- Use Media Queries to drive layout changes and to manipulate properties that can be specified through CSS styles. For example:
    - Element sizes
    - Element layout (inline, block)
    - Visibility of various elements

[1] References: http://www.w3schools.com/CSS/css_mediatypes.asp
http://ie.microsoft.com/testdrive/HTML5/CSS3MediaQueries/Default.html

- Use JavaScript events from the Application Layout API (Windows.UI.Immersive.ApplicationLayout) to drive behavior changes and to manipulate properties that cannot be specified through CSS styles. For example:
    - Scroll direction of the ListView control
    - Changing controls – going from a list of buttons to a single drop down select control

4.8.4 APPLICATION LAYOUT API

The Application Layout API exposes the application's current layout information. It allows you to:

- Register for application layout change events
- Query your current application layout state
- Unsnap your snapped app

Registering for layout changes through this API is optional, but the snapping *experience* is not. Apps that do not register for these events will be snapped anyway; your apps' window will be resized, but your app will simply not be notified.

4.8.4.1 QUERYING LAYOUT

Your app can query which layout it is currently assigned. This is particularly useful during launch or rehydration to make sure that you are loading the correct layout for your app. The following JavaScript code shows how to perform that query.

```
var appLayout =
Windows.UI.Immersive.ApplicationLayout.getForCurrentView(),
    layoutState = Windows.UI.Immersive.ApplicationLayoutState;

if (appLayout.value == layoutState.snapped) {
    // Snapped

} else if (appLayout.value == layoutState.filled) {
    // Fill

} else if (appLayout.value == layoutState.fullscreen) {
    // Full screen
}
```

4.8.4.2 HANDLING LAYOUT CHANGE EVENTS

Layout change events are fired whenever your app's layout is changed, which allows you to react appropriately to your new state (for instance, reflow your layout for the snapped state). The following JavaScript code registers an app for layout change events and provides a way for handling the new layout.

```
var appLayout =
Windows.UI.Immersive.ApplicationLayout.getForCurrentView();

function onLoad() {
    // Register application layout change events
    appLayout.addEventListener("layoutchanged", onLayoutChanged);
}

function onLayoutChanged() {
    var layoutState = Windows.UI.Immersive.ApplicationLayoutState;
    if (appLayout.value == layoutState.snapped) {
            // Handle snapped state
    } else if (appLayout.value == layoutState.filled) {
            // Handle fill state
    } else if (appLayout.value == layoutState.fullscreen) {
            // Handle full screen state
    }
}

document.addEventListener("DOMContentLoaded", onLoad, false);
```

4.8.4.3 PROGRAMMATIC UNSNAPPING

Your app can programmatically move out of the snapped state if you app has activation. However, whether your app moves into a fill or full-screen state is determined by the state of the system.

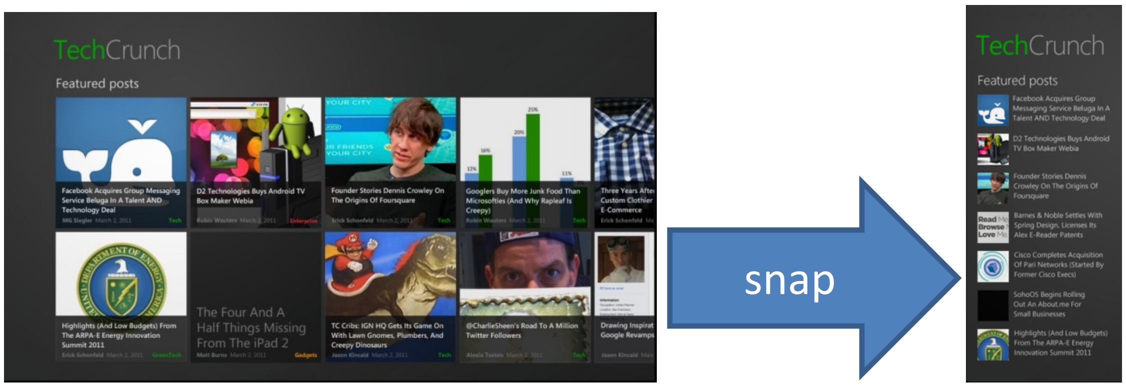
As mentioned earlier, reserve the ability to programmatically unsnap your app for situations when it is absolutely necessary to do so, such as when your app is being activated into a context that is not adequately supported in the snapped state.

The following JavaScript code demonstrates a scenario that uses programmatic unsnapping.

```
var appLayout =
Windows.UI.Immersive.ApplicationLayout.getForCurrentView(),
    programmaticUnsnapped = true;

function onSlideShowButtonClicked() {
    // Set state before calling tryUnsnap() since the order in which
    // tryUnsnap() returns and when the layoutChanged handler is triggered
    // is nondeterministic.
    programmaticUnsnapped = true;

    // Disable button so user doesn't try to repeatedly press the button
    slideShowButton.disabled = true;

    // Attempt unsnap
    var success = appLayout.tryUnsnap();

    if (!success) {
            // Reset state, clean up and show error as necessary
            programmaticUnsnapped = false;
            slideShowButton.disabled = false;
    }
}

document.addEventListener("DOMContentLoaded", function() {
    slideShowButton.addEventListener("click", onSlideShowButtonClicked,
false);
}, false);
```

4.8.4.4 RECOMMENDED SNAPPED STATE UI LAYOUTS

1. **Panning Direction**
   Take advantage of the snapped layout's "tall and skinny" aspect ratio and pan vertically. If your app changes panning direction between snapped and nonsnapped states, make sure it is clear in the UI that panning direction has changed



In this example, TechCrunch's "Featured posts" layout uses a horizontal panning ListView control (MoCo) in nonsnapped state, but converts into a vertically panning list in snapped state. Notice that the item templates have changed to sufficiently indicate that direct of panning have also changed.

The following sample code demonstrates how you can change the panning direction of a MoCo in JavaScript.

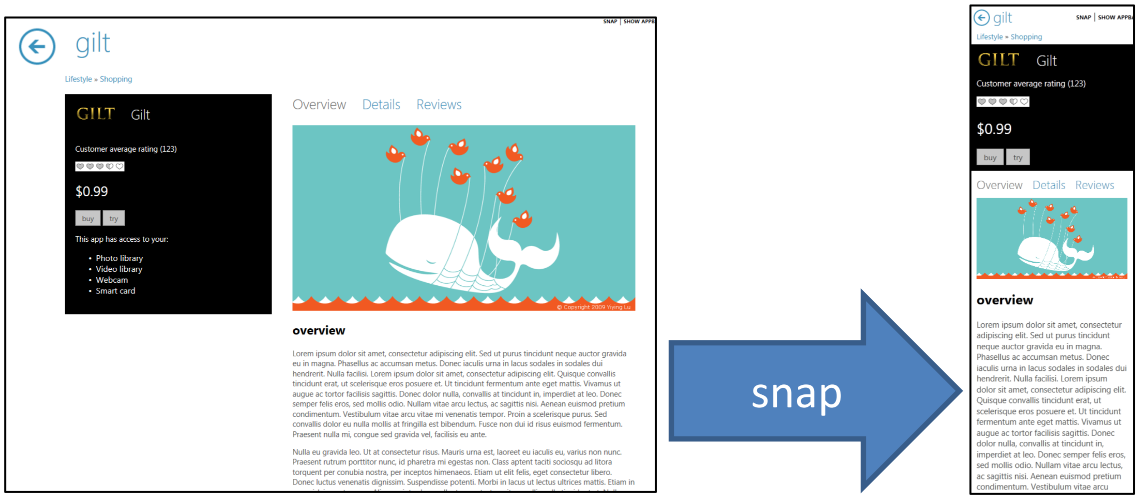
```
// There exists somewhere on the html page, an element with id 'moco' that
contains the ListView control

var moco = Win.UI.getControl(document.getElementById("moco")),
appLayout = Windows.UI.Immersive.ApplicationLayout.getForCurrentView();

document.addEventListener("DOMContentLoaded", function() {
    appLayout.addEventListener("layoutchanged", onLayoutChanged);
}, false);
```

```
function onLayoutChanged() {
    var layoutState = Windows.UI.Immersive.ApplicationLayoutState;
    if (appLayout.value == layoutState.snapped) {
            moco.layout = {type: Win.UI.ListLayout};
     } else {
            moco.layout = {type: Win.UI.GridLayout};

    }
}
```

Please see **ListView** section for more details on the control.

2. Column Layouts

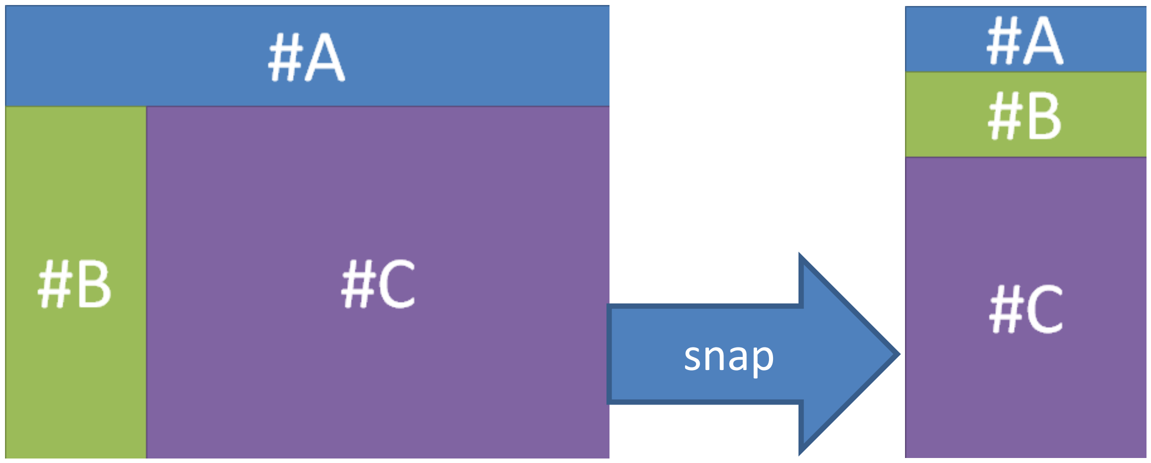
Given the narrow width of the snapped state, we recommended transitioning any multi-column layouts in the non snapped states to a single column layout when snapped.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed ut purus tincidunt neque auctor gravida eu in magna. Phasellus ac accumsan metus. Donec iaculis urna in lacus sodales in sodales dui hendrerit. Nulla facilisi. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque convallis tincidunt erat, ut scelerisque eros posuere et. Ut tincidunt fermentum ante eget mattis. Vivamus ut augue ac tortor facilisis sagittis. Donec dolor nulla, convallis at tincidunt in, imperdiet at leo. Donec semper felis eros, sed mollis odio. Nullam vitae arcu lectus, ac sagittis nisi. Aenean euismod pretium condimentum. Vestibulum vitae arcu vitae mi venenatis tempor. Proin a scelerisque purus. Sed convallis dolor eu nulla mollis at fringilla est bibendum. Fusce non dui id risus euismod fermentum. Praesent nulla mi, congue sed gravida vel, facilisis eu ante.

Nulla eu gravida leo. Ut at consectetur risus. Mauris urna est, laoreet eu iaculis eu, varius non nunc. Praesent rutrum porttitor nunc, id pharetra mi egestas non. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Etiam ut elit felis, eget consectetur libero. Donec luctus venenatis dignissim. Suspendisse potenti. Morbi in lacus ut lectus ultrices mattis. Etiam in

In this example, this app has multi-column layout in its nonsnapped state. When the app is snapped, elements of the layout effectively “stack” above one another create a single column layout.



The following sample code demonstrates how this can be achieved with the grid control via CSS:

```
<div id="grid">
    <div id="A">#A</div>
    <div id="B">#B</div>
    <div id="C">#C</div>
</div>
```

```
#grid {
    display: -ms-grid;
    -ms-grid-rows: 100px 1fr;
    -ms-grid-columns: 200px 1fr;
}

#A {
    -ms-grid-row: 1;
    -ms-grid-column: 1;
    -ms-grid-column-span: 2;
    width: 100%;
    height: 100%;
    background-color: blue;
}

#B {
    -ms-grid-row: 2;
    -ms-grid-column: 1;
    width: 100%;
    height: 100%;
    background-color: green;
}

#C {
    -ms-grid-row: 2;
    -ms-grid-column: 2;
    width: 100%;
    height: 100%;
    background-color: purple;
}

/* Until win-ui-view-state is implemented, we will use max-width to
simulate this */
@media (max-width:320px) {
    #grid {
            display: -ms-grid;
            -ms-grid-rows: 100px auto 1fr;
            -ms-grid-columns: 1fr;
    }

    #A {
            -ms-grid-row: 1;
            -ms-grid-column: 1;
    }

    #B {
            -ms-grid-row: 2;
            -ms-grid-column: 1;
    }

    #C {
            -ms-grid-row: 3;
            -ms-grid-column: 1;
    }
}
```

Please see CSS Grid Layout wiki for more details on how to use the CSS grid to control layout.

4.8.5 REFERENCES

1. Switching and Snapping Functional Spec
2. Application Layout API Document

4.9 APPLICATION LIFECYCLE AND RUNNING IN THE BACKGROUND

Windows 8 introduces a new application lifecycle model. In Windows 8 applications in the foreground (docked or in main section of the screen) are running. Applications that are not visible are suspended or not running. A suspended application stays in memory but may be removed from memory at any time if memory pressure is detected. Suspending applications that are not in the foreground helps Windows 8 achieve great battery life and system performance.

With this new application lifecycle is a set of events which indicate what part of the lifecycle the application is entering so the application can react appropriately.

Application lifecycle events

Events	Description
Suspending	Raised when the application is no longer in the foreground.
Resuming	Raised when the application is activated from a suspended state.
NotVisible	(not yet implemented)
Visible	(not yet implemented)

4.9.1 HANDLING SUSPEND

When an application is no longer visible the application receives the Suspending event if it has been registered to receive it. Once the event is raised the application has 5 seconds to handle the event. If the application does not return from the event within 5 seconds the application is terminated.

Applications will not get another chance to execute code from this stage if the system encounters memory pressure so the application should save its current state to disk so the user can be taken back to where they left off when the application activated again.

```
Windows.UI.WebUI.WebUIApplication.addEventListener("suspending",
suspendingHandler, false);

function suspendingHandler(eventArgs){
    // signaling a need to delay the suspend for this application
    var deferral = eventArgs.suspendingOperation.getDeferral();

    // start saving state
    var localFolder = Windows.Storage.ApplicationData.current.localFolder;
    var fileOperation = localFolder.createFileAsync("mystatefile",
Windows.Storage.CreationCollisionOption.replaceExisting).then(
        function(file)
        {
            // writing state into the file
            // signaling that the application is done saving the state
            deferral.complete();
    });
}
```

4.9.2 HANDLING RESUME

When an application is activated it may already be suspended in memory. If that is the case the application will be **Resumed**. When an application is resumed from a suspended state it has all of the same objects in memory it had before it was suspended.

Note: if it has been several days since the application was resumed, and taking the user back to what they were doing before does not make sense it is recommended that the resuming event reset the application to its start screen.

```
Windows.UI.WebUI.WebUIApplication.addEventListener("resuming",
resumingHandler, false);

function resumingHandler() {
    // start loading state
    var appData = Windows.Storage.ApplicationData.current;

    // the top score might have changed while the app was suspended
    var topScore = appData.roamingSettings.values.lookup("TopScore");

    // continue with other resume operations
}
```

4.9.3 APPLICATION CRASH OR HANG

If your application crashes or has been unable to process input for 10 seconds it will be terminated by the system. The application will not get a Suspending event before it is terminated.

The next time the application is launched the application can check for a “start fresh” bit that is the indication to the application that it was terminated unexpectedly. Applications should consider disregarding any stored state in this case in the event that the stored state caused the crash. This can be checked upon receiving the activation event using the reason property on the activation event args.

4.9.4 BACKGROUND AUDIO

In general, applications do not get to execute code in the background. However, if the application needs to play audio it can be allowed to execute when the application is not visible on the screen.

To enable this developers must add the following to the application manifest:

```
  <Extensions>
    <Extension Category="windows.backgroundTask" EntryPoint="MyApp.Audio">
      <BackgroundTasks>
           <Task Type="audio"/>
      </BackgroundTasks>
    </Extension>
  </Extensions>
```

To play audio in the background developers must declare an audio stream type of “Media,” “Narration,” or “Communications.” This can be done via an attribute of an audio/video tag, like this:

```
<audio src="song.mp3" ms-AudioCategory="Multimedia" controls />
```

An audio category can also be selected directly via the Windows Audio Streaming API’s (WASAPI).

The complete list of categories is:

```
typedef enum
{
    Multimedia,      // Music, Streaming audio, Video with audio
    Communications, // VOIP, chat, phone call
    Alerts,          // Alarm, ring tones, notifications
    GameMedia,       // Background music for games (non-effect game sounds)
    SoundEffects,    // Sound effects, clicks, dings, gun shot
    Narration,       // Screen reader, eBook reader
    Other            // Uncategorized streams
} AudioCategory;
```

Each audio output stream has a stream type indicating the type of audio being played. Only one audio stream per type is allowed to play in the background at any given time – but if multiple audio types are playing, they might all play simultaneously (eBook while listening to music).

Note: when the audio playing application goes to the background it will not get a Suspending event.

4.10 SEARCH

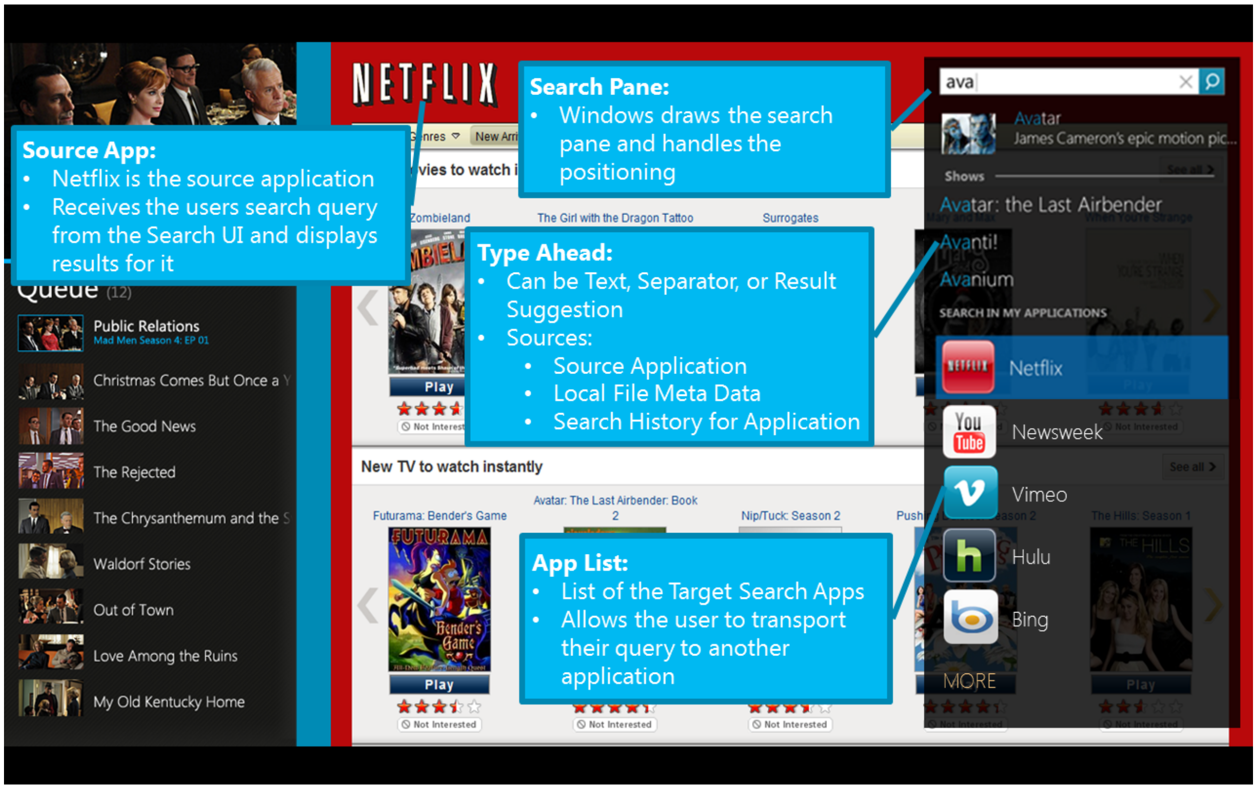
The search application contract enables developers to use the Windows Search UI to provide search functionality for their application. This contract has two required parts:

- The source application contract for creating an in-app search experience.
- The target application contract to handle being activated with a specific query.

The source application is the active MoBody application that uses the Windows Search UI to provide an in-app search experience. For this experience, a source app can use Type Ahead to display search suggestions as the user enters their search query. Type Ahead allows these search suggestions to be supplied by the application or by Windows, which uses search history or local file metadata. The source application handles the submission of the user's query and displays the search results. The source application fulfills its contract by using the WinRT **SearchPane**, which provides search-related properties and events, such as **QuerySubmitted**.

A target application is an application that can receive the search query even though it is not running in the MoBody. Target applications are displayed in the Windows Search UI after search suggestions. If a user selects one of these applications, Windows launches the selected application directly into the MoBody so that it shows search results for the user's query. A target application fulfills the target contract by registering through its manifest and adding support for being activated to its Search mode (the application is launched to its search results view for the query).

While Search itself is an optional contract, any search application has to fulfill both source and target contracts to participate in the search experience. If an application doesn't participate in Search, the Windows Search UI doesn't show the app in the search pane's app list.



Source App:
Netflix is the source application
Receives the users search query from the Search UI and displays results for it
Search Pane:
Windows draws the search pane and handles the positioning
Type Ahead:
Can be Text, Separator, or Result Suggestion
Sources:
Source Application
Local File Meta Data
Search History for Application
App List:
List of the Target Search Apps
Allows the user to transport their query to another application
ava
Avatar
Avanti!
Avanium
SEARCH IN MY APPLICATIONS
Netflix
Newsweek
Vimeo
Hulu
Bing
MORE
New TV to watch instantly
Queue

Figure 1: Example of an Application that participates in the Search contract

4.10.1 OVERVIEW OF THE SEARCH CONTRACT—THE SOURCE APP CONTRACT

The source application (the active MoBody application), uses the MoSh Search UI and Type Ahead to provide an in-app search experience using the **SearchPane** object provided by the Windows Runtime. The **SearchPane** should be configured for the application before the communication between the **SearchPane** and the application begins. For a better East-Asian experience, Windows will provide linguistic alternatives so applications can perform searches based on a user's phonetic input, without the need to wait for the user to convert their input into Chinese or Japanese characters.

4.10.1.1 THE SEARCH SYSTEM FLOW

1. The user launches a modern application that supports Search.
2. Upon being launched and shown in the MoBody, the application obtains the **SearchPane** WinRT object and sets up the relevant events and properties for its search experience.

3. At a minimum, the application needs to register to handle the **QuerySubmitted** event so that it can show the search results for a user's query. If the application wants to provide its own Type Ahead, the application will also need to handle the **SuggestionsRequested** event. If the application uses Windows to provide Type Ahead, Windows will use either search history and/or **LocalContentSuggestions**, depending on how the **SearchPane** is configured.
4. When the user opens the Search Pane via the Search Charm and begins typing, the **SuggestionsRequested** event is called. The application has the ability to provide its own Type Ahead items that correspond to the user's current search query. If the application wants to provide a word wheel search experience it can use the **QueryChanged** event.
5. When the user submits their search, the application's **QuerySubmitted** event is raised. The application retrieves and displays search results for the query it receives in this event.

4.10.1.2 WINRT API DETAILS

SearchPane Static Methods

Method	**Description**
GetForCurrentView	Enables the developer to obtain the SearchPane object for the apps current view.

SearchPane Object

Property	**Description**
SearchHistoryContext	Enables the developer to have more flexibility over how previous searches are stored. If there are two different search scopes, the developer can have previous searches stored on a per scope basis. This property does not have to be specified.
PlaceholderText	The string that appears in the Action Space Search Box before any character is entered into the Search Box.
QueryText	Sets or retrieves the Search Control's Query Text.
Visible	Determines whether the Search Control is visible.
LanuageTag	Specifies the user's locale.
SearchHistoryEnabled	Specifies whether Windows tracks search history and uses it to provide suggestions for the Search control.
Events	**Description**
VisibilityChanged	Raised when the Seach Pane is shown or hidden.
QueryChanged	Raised when the user updates their search query.
QuerySubmitted	Raised when the user initiates a search.
SuggestionsRequested	Raised whenever the application can provide suggestions for a user's query input.
Method	**Description**

Show	Shows the Search Pane in the minimized mode.
SetLocalContentSuggestionSettings	Specifies settings for Windows automatically providing Search Suggestions from the user's Local Files.

LocalContentSuggestionSettings Object

Property	**Description**
Enabled	Allows the developer to specify if it wants Windows to automatically provide Search Suggestions from the user's Local Files
Locations	A list of Storage Folders which represent local libraries or folders to retrieve LocalContentSuggestions from.
AqsFilter	Filters the files of the specified scopes to further restrict the LocalContentSuggestions.
PropertiesToMatch	Determines which properties are used for LocalContentSuggestions.

4.10.2 THE TARGET APP CONTRACT

When an application registers for the Search contract through its manifest, the application appears in a list provided by the MoSh Search UI when the user issues a query in other applications. When an application is selected from the app list, it becomes the active MoBody application and shows the search results for the user's query. The target app registration provides details about how the application can be launched directly to its search view either by specifying an **HTML page** or an **ActivateableClassID**.

The target app Search contract is executed as follows:

1. The developer adds the search registration markup to the app's manifest. This registration includes details on how to launch the application directly to its Search mode.
2. The user installs the application.
3. Upon install, Windows stores the search registration details from the app's manifest in the contract data store.
4. The user clicks the Search Charm to display the Search Pane.
5. The Search Pane uses the contract data store registrations to populate the Search Pane UI with all the applications that can receive the query.
6. After the user types a query and selects an application in the Search Pane, Windows uses the registration details in the contract data store to launch the application in its Search mode.
7. Windows uses the WinRT **Activation** object to send the search query to the application.

The following code shows the manifest registration for the Search Target contract.

```
<Extension Category="windows.search" StartPage="search.html">
</Extension>
```

Activation Context WinRT Object:

Activation Context Object	
Property	**Description**
QueryText	The user's query.
LanguageTag	The user's locale.

4.10.3 SAMPLE JS CODE

The following example shows the minimum required code for implementing the source and target Search contracts.

```
Required Manifest Entry:
<Extension Category="windows.search" StartPage="search.html">
</Extension>

Required JavaScript Code:
// Supporting Search Contract Activation
Windows.UI.WebUI.WebUIApplication.addEventListener("activated",
scenario1SearchActivated, false);

function scenario1SearchActivated(e) {
    // Search activation
    if (e.contractId === "Windows.Search") {
        if (e.queryText === "") {
            // Navigate to your landing page since the user is pre-scoping
to your app
        } else {
            // Display results in UI for e.queryText
            showResultsForQuery(e.queryText, e.languageTag);
        }
    } else { // Non Search Activation
        // Other Activation Code
    }
}

// Support receiving a search query while running as the main application
Windows.UI.Immersive.Search.SearchPane.getForCurrentView().addEventListene
r("querysubmitted", querySubmitted, false);

function querySubmitted(e) {
    // Display results in UI for e.queryText
    showResultsForQuery(e.queryText, e.languageTag);
}

function showResultsForQuery(query, locale) {
    // Draw results in UI
}
```

The following example shows how to enable LocalContentSuggestions from the music library.

```
var localSuggestionSettings = new
Windows.UI.Immersive.Search.LocalContentSuggestionSettings();

localSuggestionSettings.enabled = true;
localSuggestionSettings.locations.append(Windows.Storage.KnownFolders.musi
cLibrary);
localSuggestionSettings.aqsFilter = "kind:=music";


Windows.UI.Immersive.Search.SearchPane.getForCurrentView().setLocalContent
SuggestionSettings(localSuggestionSettings);
```

The following example shows the code required for an application to provide search suggestions.

```
Windows.UI.Immersive.Search.SearchPane.getForCurrentView().addEventListene
r("suggestionsrequested", handleSuggestionRequest, false);

// Provide suggestions from app defined list
function handleSuggestionRequest(e) {
    var query = e.queryText.toLowerCase();
    var suggestionRequest = e.request;
    var suggestionList = ["Shanghai", "Istanbul", "Karachi", "Delhi",
"Mumbai", "Moscow", "São Paulo", "Seoul", "Beijing", "Jakarta", "Tokyo",
"Mexico City", "Kinshasa", "New York City", "Lagos"];

    for (var i = 0, len = suggestionList.length; i < len; i++) {
       if (suggestionList[i].substr(0, query.length).toLowerCase() ===
query) {

    suggestionRequest.searchSuggestionCollection.appendQuerySuggestion(
suggestionList[i]);
          if (suggestionRequest.searchSuggestionCollection.size === 5) {
               break;
          }
       }
    }
}
```

4.10.4 REFERENCES

1. Search API Doc
2. Search Functional Spec

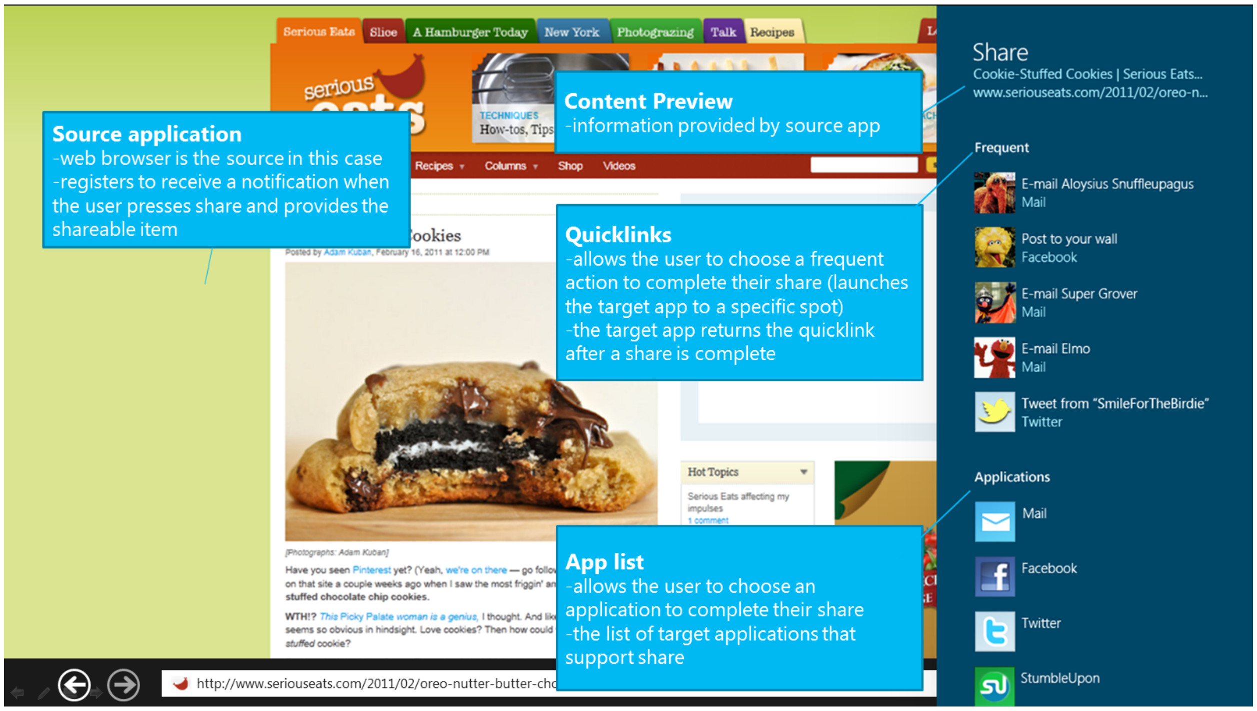
4.11 SHARE

Sharing enables users to share content from their current application with another application. Developers of any Windows application can utilize the Windows share functionality to enable their users to share content. Apps that provide shareable content are called source applications. The source application is the active application in the MoBody. An example source application is a magazine reader.

Developers of sharing applications can utilize the share functionality so that users can share to their application from any source application. Apps that receive shareable content and make it visible to other users are called target applications. The target application is displayed in a flyout window. An example target application is Facebook.

When the user chooses the Share Charm from the Windows Edgy bar, the Share flyout opens in response:

1. Source applications provide the shareable item, potentially in multiple formats. The first format is displayed in the content preview. The shareable item is also used to determine which QuickLinks and applications to display in the list. Items are excluded if they cannot support at least one of the formats for the shareable item.
2. Applications and quicklinks are displayed based on whether their package manifest indicates that they support sharing. Quicklinks are displayed based on data passed back by target applications after the user has used them to share.
3. After a user chooses a quicklink or application from the list, the target app flyout opens. Windows launches the correct target application and the target application displays UI for sharing. The target application reads the shareable item from the source application.



Source application
-web browser is the source in this case
-registers to receive a notification when the user presses share and provides the shareable item

Content Preview
-information provided by source app

Quicklinks
-allows the user to choose a frequent action to complete their share (launches the target app to a specific spot)
-the target app returns the quicklink after a share is complete

App list
-allows the user to choose an application to complete their share
-the list of target applications that support share

Share
Cookie-Stuffed Cookies | Serious Eats...
www.seriouseats.com/2011/02/oreo-n...

Frequent
E-mail Aloysius Snuffleupagus
Mail
Post to your wall
Facebook
E-mail Super Grover
Mail
E-mail Elmo
Mail
Tweet from "SmileForTheBirdie"
Twitter

Applications
Mail
Facebook
Twitter
StumbleUpon

Figure 2: A source app participating in the sharing contract

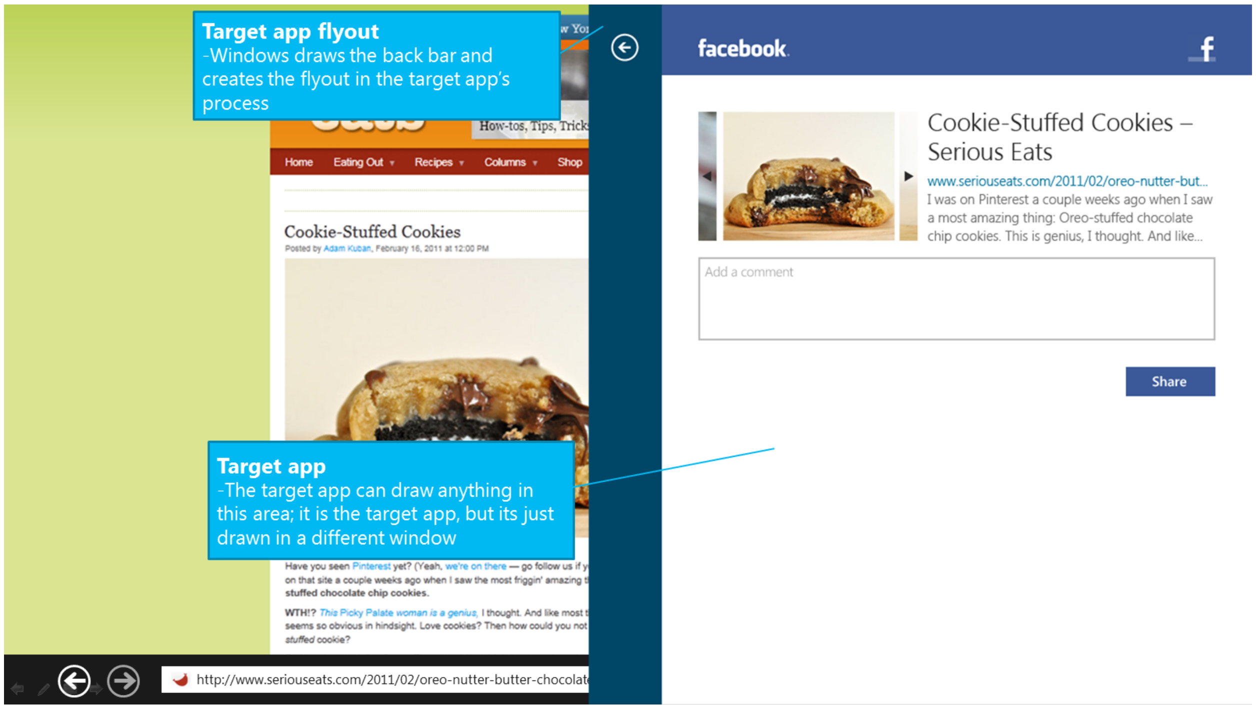

Target app flyout
-Windows draws the back bar and creates the flyout in the target app's process
Target app
-The target app can draw anything in this area; it is the target app, but its just drawn in a different window
facebook
Cookie-Stuffed Cookies – Serious Eats
www.seriouseats.com/2011/02/oreo-nutter-but...
I was on Pinterest a couple weeks ago when I saw a most amazing thing: Oreo-stuffed chocolate chip cookies. This is genius, I thought. And like...
Add a comment
Share
Cookie-Stuffed Cookies
http://www.seriouseats.com/2011/02/oreo-nutter-butter-chocolat

Figure 3: A target app participating in the sharing contract

4.11.1 OVERVIEW OF THE SHARE SOURCE APP CONTRACT

By default, when the user shares data from an application, the system gives the user an option to take a screenshot of the source application and share it with a target application. If the application wants to override the system default behavior and implement the contract it has to:

1. Create **DataTransferManager** object & handle **DataRequested** notification.
2. If you're retrieving data synchronously, set the **Data** or invoke the **FailWithDisplayText** method on the **DataRequestedEventArgs** object before the timeout.
3. If you're retrieving data asynchronously, then retrieve a **DataRequestDeferral** object and call Complete after the data is set or after there's an error getting the data.

Property	Description
Title (required)	A string for title of sharable content
Description (optional)	A string for brief description of content

4.11.1.1 THE SHARE SYSTEM FLOW – SOURCE APP

1. The user opens the source application. The source application registers to support Share by creating a **DataTransferManager** object.

2. When the user invokes Share from the Windows Edgy bar, the system takes a screenshot that can be used if the user selects it from the default screen. If the application has registered for sharing, the system instead asks the application for the shareable item.
3. The system uses the formats of the shareable item to filter the quicklinks and app list and to display the Content Tile.
    a. Note: If the source app is going to use delayed rendering of FileItems, the system uses the file extensions provided in the **FileTypes** property to filter the quicklinks and app list
4. The user chooses the target app. The system adds the applications to the OS Task completion list so they cannot be killed while participating in the sharing flow. The **TargetApplicationChosen** notification is sent so the source application can record the app for business intelligence purposes.
5. *<See steps the next section for information about what the flow after the user chooses a target application.>*

4.11.1.2 WINRT API DETAILS

The source contract for the Sharing and SendTo flows utilize the same WinRT objects. For charms invocation, the source application doesn't know whether it was invoked for Sharing or Send To. For programmatic invocation, the source application has to choose which UI to display.

Windows.UI.Immersive.DataTransfer.DataTransferManager Static Object – used for programmatic invocation only [optional]

Methods	**Description**
ShowShareUI	Programmatic invocation of the Share flow. Most source applications should not need to use this because they can rely on the charms invocation.
ShowSendUI	Programmatic invocation of the Send flow (in the Connect charm). Most source applications should not need to use this because they can rely on the charms invocation.
GetForCurrentView	Returns DataTransferManager object associated with the current window.

Windows.UI.Immersive.DataTransfer.DataTransferManager Object

Events	**Description**
DataRequested	This notification occurs when the user presses the Share or Connect charm (the source application intentionally does not know which charm the user pressed). The app needs to respond within the timeout by calling a method or setting the property on the *DataRequest* object retrieved through the *DataRequestedEventArgs* object.
TargetApplicationChosen	This notification occurs when the user chooses a target application in the Share or Connect charm. The app can use this to record information for

	business intelligence purposes about which applications are used to complete actions.

Windows.UI.Immersive.DataTransfer .DataRequestedEventArgs Object	
Property	**Description**
Request	Allows developer to get the *DataRequest* object to either set the data or fail with display text.

Windows.UI.Immersive.DataTransfer.DataRequest Object	
Methods	**Description**
FailWithDisplayText	Needs to be called before timeout in response to the event notification, *DataRequested*, to opt-out of Sharing or Send To given the current state of the source application and provide an error string instead.
GetDeferral	Returns *DataRequestDeferral* object which is necessary in order to set data for share that is retrieved via an asynchronous operation.
Property	**Description**
Data	Needs to be set before timeout in response to the event notification, *DataRequested*, to provide a Data Package for Share or Send.

Windows.UI.Immersive.DataTransfer.DataRequestDeferral Object	
Methods	**Description**
Complete	Is used in scenarios when data for share is retrieved via an asynchronous operation. After the data is set or an error occurs, call Complete.

Windows.UI.Immersive.DataTransfer.TargetApplicationChosenEventArgs Object – used for gathering business intelligence about the target application only [optional]	
Property	**Description**
ApplicationName	After the event notification, *TargetApplicationChosen*, this property is set with the target application's name.

4.11.2 OVERVIEW OF THE SHARE TARGET APP CONTRACT

To be a target application for Share, an application must:

1. Register support for the **ShareTarget** extension in package manifest and include at least one supported file extension or data package format in application manifest (hint: file extensions must come first!)
2. Handle Share Activation & use the *Data* provided on the **ShareTargetActivationContext** object. **ShareTargetActivatedEventArgs**, which is passed through onActivate(), contains a property called *ShareTargetActivationContext* which is of type **ShareTargetActivationContext**.
3. For an extended share only, call **ReportExtendedShareStatus** once the user has committed the share task and the app is ready to be added to the share progress list.
    a. Optionally report additional status to optimize resource usage of the system.
    b. Optionally call **SetExtendedShareDisplayText** to be shown on the app tile in the progress list.
    c. Optionally report a catastrophic failure with **ReportExtendedShareError**.
4. Call **ReportCompleted** when the share is complete and return a **QuickLinkInfo** object if the share was successful

4.11.2.1 SHARE SYSTEM FLOW – TARGET APP

1. The user installs the target application. The target application's package manifest declares that it supports sharing.
2. The system copies this information and stores it so it can be later queried to display applications in the share flyout.
3. *<See steps above in the system flow as it pertains to the source application>*
4. When the user chooses the target application, the system will activate that application via its application interface and using the details it provided during registration to launch it directly to its share view
    a. While the application is being activated, the system will display a splash screen for the application
5. The system will add the target application to the OS Task completion list so it cannot be killed while part of the sharing flow or when completing a share in the background
6. If the target application handles extended shares—usually the case if they upload content types that can take more than a few seconds—they report status as they go:
    a. They report when the user presses the Submit/Share/Post/Upload button and thus if the user taps outside of the flyout, the system will hide the window vs. destroy it
7. When the target application is finished, it calls ReportCompleted and the system stores the quicklink. At this point, the system removes it from the OS Task completion list.

Note: There are optimizations the target can do to preserve battery life and these are covered in more detail in the Sharing Platform functional spec. They only apply to target applications that support extended share scenarios.

The following example shows how an application registers for sharing in its package manifest.

```
<Extension Category="windows.shareTarget" StartPage="share.html">
          <ShareTarget>
            <SupportedFileTypes SupportsAnyFileType="false">
              <FileType>.png</FileType>
              <FileType>.customFileType</FileType>
            </SupportedFileTypes>
            <DataPackageFormat>Uri</DataPackageFormat>
            <DataPackageFormat>Text</DataPackageFormat>
          </ShareTarget>
</Extension>
```

4.11.2.2 WINRT API DETAILS

Windows.UI.Immersive.ShareTarget.ShareTargetActivationContext Object

Property	**Description**
Data	A Modern Data Package that contains the data passed by the source application.
QuickLinkId	The string for the quicklink that was selected by the user. This is just passing back to the application what it original set as the *TargetId* in the *QuickLinkInfo* object.
Methods	**Description**
ReportCompleted	The target application should call this after the share has completed successfully or unsuccessfully. The complete call closes the UI and effectively shuts down the application. For a **quick share**, this should be called immediately after the user presses 'share'. A quick share is a task where it's reasonable for the user to view the UI until the share completes, such as posting a message to Twitter. For an extended share, this will be called after the final data transfer/upload is complete, which is likely quite a while after the user presses 'share'. An **extended share** is a task where it's unreasonable to make the user wait and view the share UI until the share completes, because it will take more than 5 seconds, such as uploading photos to Flickr.
ReportExtendedShareError	The target application should call this after the share has completed unsuccessfully and it has the option of returning a string that will be displayed in the UI. Generally, the application should report an error only if something unrecoverable has happened

RemoveThisQuickLink	The target application should call this if it needs to remove a quicklink because it has become out of date and no longer relevant to the user. This will completely remove that quicklink from the user's history. If an application just wants to update a quicklink, it can do so by updating information in the QuickLinkInfo object it returns.
ReportExtendedShareStatus	If the target application supports **extended share** scenarios, then it should use this method to report status to the share platform. The application needs to report values defined in the *ExtendedShareStatus* enum: • *Started* – This is *required* for extended shares to behave correctly in the UI model. The application should report this after the user has committed the share task. This puts the app in the share progress list and enables a user to close the window without shutting down the application and thus stopping the share task. • *DataExtracted* – This is *optional* for extended shares to optimize resource usage of the system. Applications should report this if they have finished extracting data from the Data Package. • *SubmittedToBackgroundManager* – This is *optional* for extended shares to optimize resource usage of the system. Applications should report this if they have handed off the remainder of the share task to the Background Manager. An example is the Upload APIs.
SetExtendedShareDisplayText	The target application can call this while it is running to modify the string that is displayed in the background progress window. This is how an application can call the user's attention to take action in their application.

Windows.UI.Immersive.ShareTarget.QuickLinkInfo Object

Property	**Description**
Title	The title string to display in the UI for the quicklink.
Description	In M2, there was a description string displayed in the UI.
Thumbnail	This is a string that represents a thumbnail resource, either as a file or resource reference. (e.g. "c:\pictures\mythumbnail")
TargetId	This is a string that your application will be passed back when a user chooses this QuickLink in the future.

SupportedDataPackageFormats	A list of supported data package formats that this quicklink supports.
SupportedFileExtensions	A list of supported file extensions that this quicklink supports.

4.11.3 SAMPLE JAVASCRIPT CODE

The following code runs on the source app.

```
// Simple sharing scenario
var dtm =
Windows.UI.Immersive.DataTransfer.DataTransferManager.getForCurrentView();
dtm.addEventListener("datarequested", function (ev) {
    // Populate the data package properties
    ev.request.data.properties.title = "Title"; // required
    ev.request.data.properties.description = "Description"; // optional

    // Populate data package format(s), this is sharing a text item
    ev.request.data.setText("http://www.buildwindows.com");
});
```

The following code runs on the target app.

```
// This is the sharing contract entry point
var shareCtx;
function onShareTargetActivated(e) {
    shareCtx = e.shareTargetActivationContext;
}

function onShareSubmit(e) {
    var dp = shareCtx.data; //get the shareable item
    if
(dp.contains(Window.ApplicationModel.DataTransfer.StandardDataFormats.stor
ageItems)) { //if the data is a set of files
        dp.getStorageItemsFormatAsync().then(function (storageItems) {
            for (var i = 0, l = storageItems.length; i < l; ++i) {
                var blob =
windows.msBlobHelper.createBlobFromStorageItem(storageItems[i]);
                uploadToServer(blob);
            }
            shareCtx.reportCompleted(null); //report successful
completion, but do not include a quicklink
        });
    }
}
```

4.11.4 REFERENCES

1. Share M3 User Experience Functional Spec

2. Share M3 Platform Functional Spec
3. Share API Review Doc
4. Building an application using share – UX guidance
5. Code Samples

4.12 SETTINGS CHARM - APPLICATION SETTINGS

The settings charm is great at providing users with fast, in-context access to settings that affect the current immersive app experience. These settings always include

- Commonly accessed system settings applicable in any context
- Settings and information applicable to the application that the system exposes

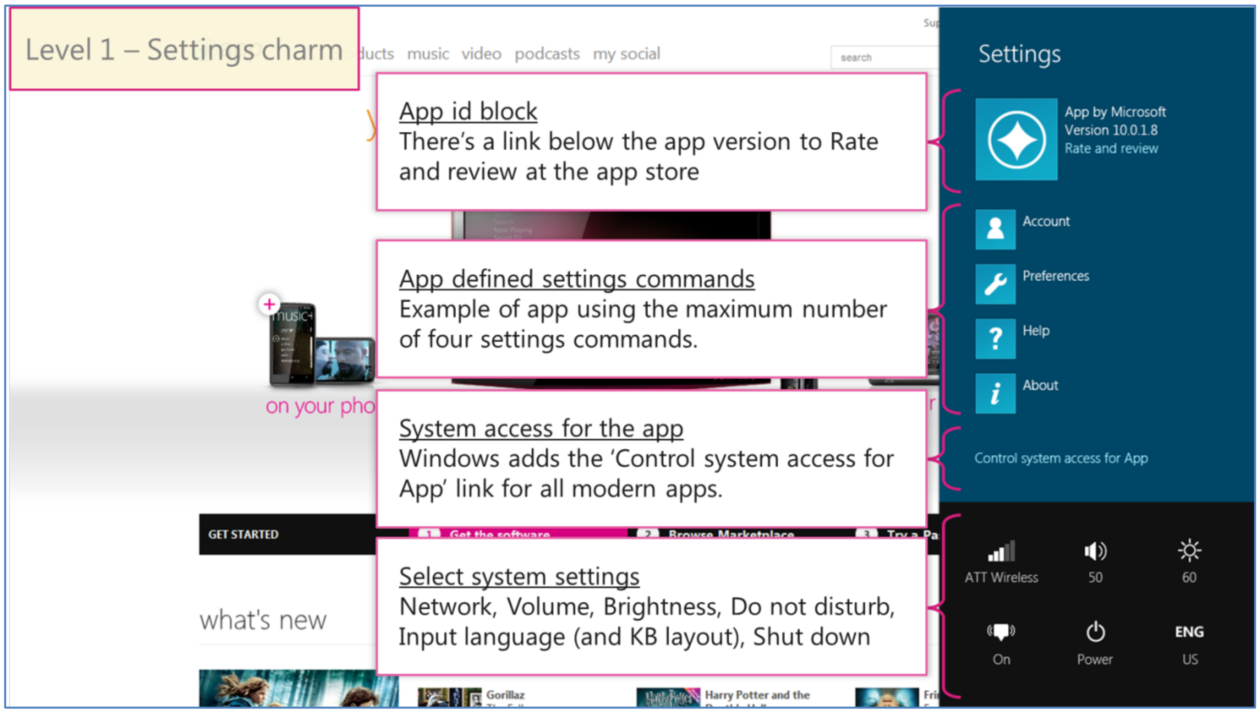
If the application participates in the settings contract they also include application specified settings commands that launch an application defined settings UI.

Exposing settings commands in the settings charm gives developer a unique chance to integrate with Windows top level UI and provide access to their settings UI at all time without scarifying application real estate or having to build navigation to and from a setting page.

4.12.1 USER EXPERIENCE

With an immersive application running the user can invoke the settings charm via the charms bar or with the keyboard by pressing key + I. The settings charm window appears docked on the right edge of the screen and comprises two panes:

- The app pane, occupying the top part of the settings charm window, contains information and option scoped to the current immersive app or supported system surface
  - App ID block provides identification for the app or system surface
  - If the app has opted in the settings contract the app settings commands provide access to second level UI where the app settings are exposed
  - The system access link provides information and control on the system capabilities and devices that the app can use
- The system pane, occupying the bottom part of the settings charm window, provides access to settings and functionality that are scoped to the whole system



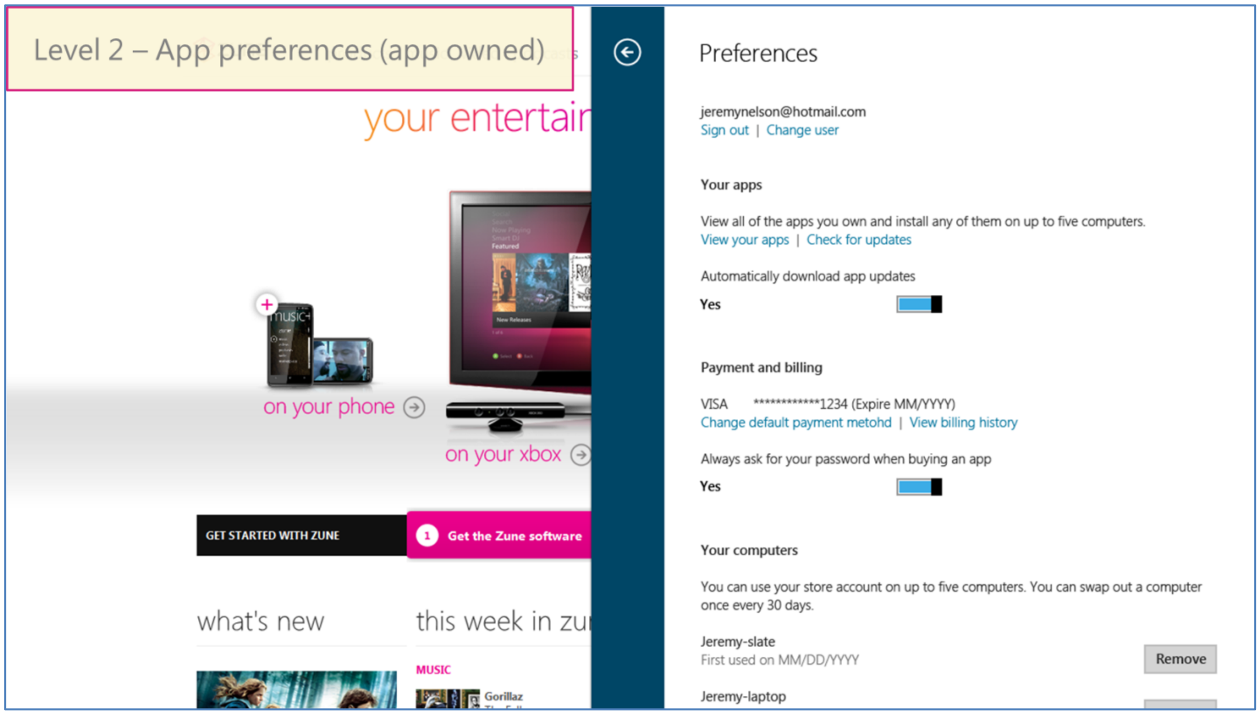
App id block
There's a link below the app version to Rate and review at the app store

App defined settings commands
Example of app using the maximum number of four settings commands.

System access for the app
Windows adds the 'Control system access for App' link for all modern apps.

Select system settings
Network, Volume, Brightness, Do not disturb, Input language (and KB layout), Shut down

When the user selects one of the app defined settings commands, the settings charm closes and the app defined settings UI associated with that command appears.



Preferences

jeremynelson@hotmail.com
Sign out | Change user

Your apps

View all of the apps you own and install any of them on up to five computers.
View your apps | Check for updates

Automatically download app updates
Yes

Payment and billing

VISA ************1234 (Expire MM/YYYY)
Change default payment metohd | View billing history

Always ask for your password when buying an app
Yes

Your computers

You can use your store account on up to five computers. You can swap out a computer once every 30 days.

Jeremy-slate
First used on MM/DD/YYYY
Remove

Jeremy-laptop

4.12.2 OVERVIEW OF THE APPLICATION SETTINGS CONTRACT

For the settings charm to expose application settings commands the application must implement the settings contract. Thus, for an application, participating in the settings contract means:

- the application has specified some settings commands to be exposed in the settings charm
- the application shows the settings UI it has defined in response to the user selecting any of these settings commands

Adding settings command is done by

1. instantiating a **SettingsPane** object
2. adding settings commands to the **SettingsPane** object **applicationCommands** vector.
    a. These settings command can be predefined. “Account”, “Preferences”, “About”, “Help” and “Terms of use” are available and accessible as static properties off the **SettingsPane** object
    b. These command can be custom by specifying a name, and ID and an image for the settings command
    c. Adding settings command is typically done at application start but can also be done dynamically in reaction to the **settingsopening** event.
3. An event handler must be specified in the constructor for the settings commands

Showing the settings UI requires

1. That the application implements the event handlers specified when adding the settings command
2. The appropriate settings UI gets shown in reaction to the event associated with the settings command

4.12.3 APPLICATION SETTINGS UI RECOMMENDATIONS

Settings commands

- Use predefined settings commands as much as possible
- Expose the same settings commands regardless of app context
- Contextualize the content if you need, not the type of UI launched
- Always open a second level UI from a settings command (no top level command without visual feedback)
- Don’t use a settings command to provide commands used as part of a typical app workflow (e.g. changing the paint color in a Painter app)
- Don’t use a settings command to navigate to a destination in the app (causing the user to have to navigate back)

Settings UI

While the surface in which the settings UI is displayed is at the discretion of the app there is a strong preference for a light dismiss surface visually coming from the settings charm*.

* We will be providing a WWA control to host settings UI as a M3 DCR.

1. Make settings instant commit where possible
2. Avoid lateral navigation: use back button navigation as provided
3. Aim for simplicity consistency, align with other apps
4. Use the style sheet where possible and avoid custom controls
5. Keep it flat: Avoid more than 2 levels for the panel
6. Lay out content top to bottom in a single column
    1. Decrease the number of settings where possible, keep it simple
    2. Use a second level or an expand/collapse model as appropriate for the content
    3. Combine less important content into a single entry point so more important content can be split across the other entry points
7. Consider going full screen when an explicit dismiss is required, but return to the settings panel upon explicit dismiss

4.12.4 SAMPLE JAVASCRIPT CODE

The following code demonstrates adding Preferences and Help to the settings charm and capturing the associated events:

```
function addSettingsCommands() {
var settingsSample = Windows.UI.Immersive;
var vector = null;
vector = settingsSample.SettingsPane.applicationCommands;

//Ensure no settings commands are currently specified in the settings
charm
if (vector == null) {
    alert("vector is null");
    }
vector.clear();

var commandPreferences;
try {
     commandPreferences = new
settingsSample.SettingsCommand(settingsSample.SettingsCommand.preferences,
scenario2OnPreferences);
    } catch (e) {
      alert("Error calling
settingsSample.SettingsCommand(settingsSample.SettingsCommand.preferences,
scenario2OnPreferences): " + e);
    }
try {
     vector.append(commandPreferences);
    } catch (e) {
        alert("Error calling vector.append(commandPreferences): " + e);
    }

var commandHelp;
try {
     commandHelp = new
settingsSample.SettingsCommand(settingsSample.SettingsCommand.help,
scenario2OnHelp);
    } catch (e) {
        alert("Error calling
settingsSample.SettingsCommand(settingsSample.SettingsCommand.help,
scenario2OnHelp): " + e);
    }

try {
     vector.append(commandHelp);
    } catch (e) {
            alert("Error calling vector.append(commandHelp): " + e);
        }
    }

function scenario2OnPreferences() {
    alert("Preferences settings command selected.");
}

function scenario2OnHelp() {
    alert("Help settings command selected.");
}
```

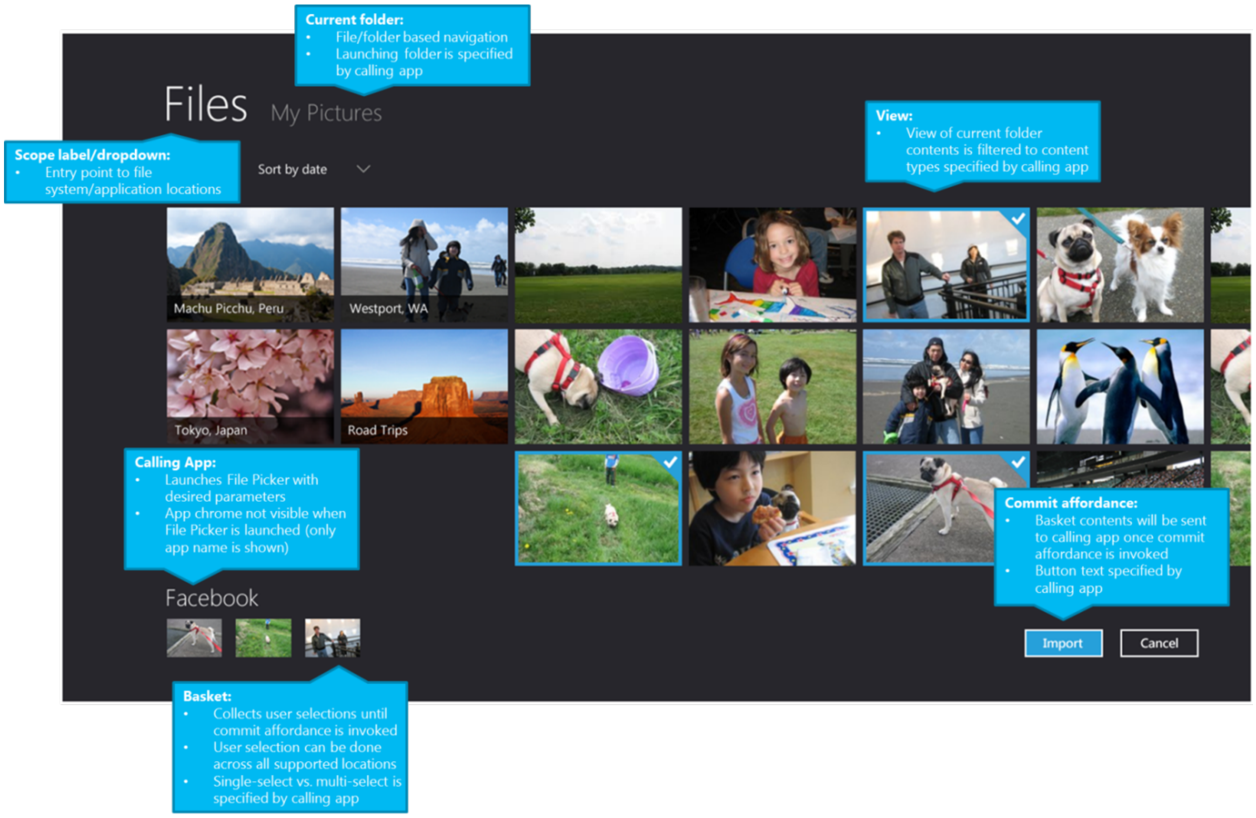
4.12.5 REFERENCES

1. Settings Charm - M3 PM Spec.docm
2. Settings API Review Doc
3. Settings DDI doc
4. Settings UI rollout presentation
5. SettingsTestApp

4.13 FILE PICKER

The **File Picker** is a Windows control that primarily provides open, save, and location picking functionality to applications. The **File Picker** APIs provide a way for developers to access files and folders across the local system, HomeGroup, devices, UNC, and applications. There are three primary modes: File, Save, and Folder.

File Mode:



Current folder:
- File/folder based navigation
- Launching folder is specified by calling app

Scope label/dropdown:
- Entry point to file system/application locations

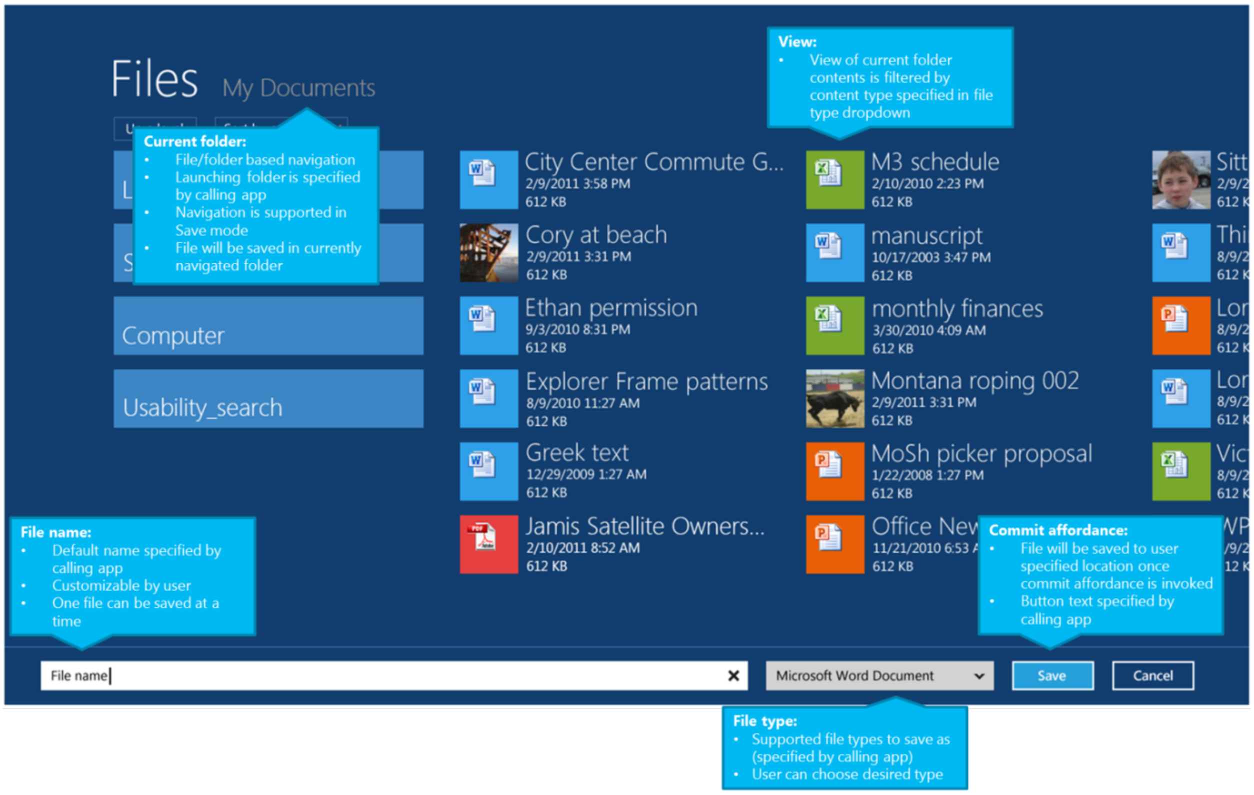
View:
- View of current folder contents is filtered to content types specified by calling app

Calling App:
- Launches File Picker with desired parameters
- App chrome not visible when File Picker is launched (only app name is shown)

Commit affordance:
- Basket contents will be sent to calling app once commit affordance is invoked
- Button text specified by calling app

Basket:
- Collects user selections until commit affordance is invoked
- User selection can be done across all supported locations
- Single-select vs. multi-select is specified by calling app

Save Mode:


Files My Documents
Computer
Usability_search
City Center Commute G... 2/9/2011 3:58 PM 612 KB
Cory at beach 2/9/2011 3:31 PM 612 KB
Ethan permission 9/3/2010 8:31 PM 612 KB
Explorer Frame patterns 8/9/2010 11:27 AM 612 KB
Greek text 12/29/2009 1:27 AM 612 KB
Jamis Satellite Owners... 2/10/2011 8:52 AM 612 KB
M3 schedule 2/10/2010 2:23 PM 612 KB
manuscript 10/17/2003 3:47 PM 612 KB
monthly finances 3/30/2010 4:09 AM 612 KB
Montana roping 002 2/9/2011 3:31 PM 612 KB
MoSh picker proposal 1/22/2008 1:27 PM 612 KB
Office New 11/21/2010 6:53 612 KB
File name
Microsoft Word Document
Save
Cancel

View:
- View of current folder contents is filtered by content type specified in file type dropdown

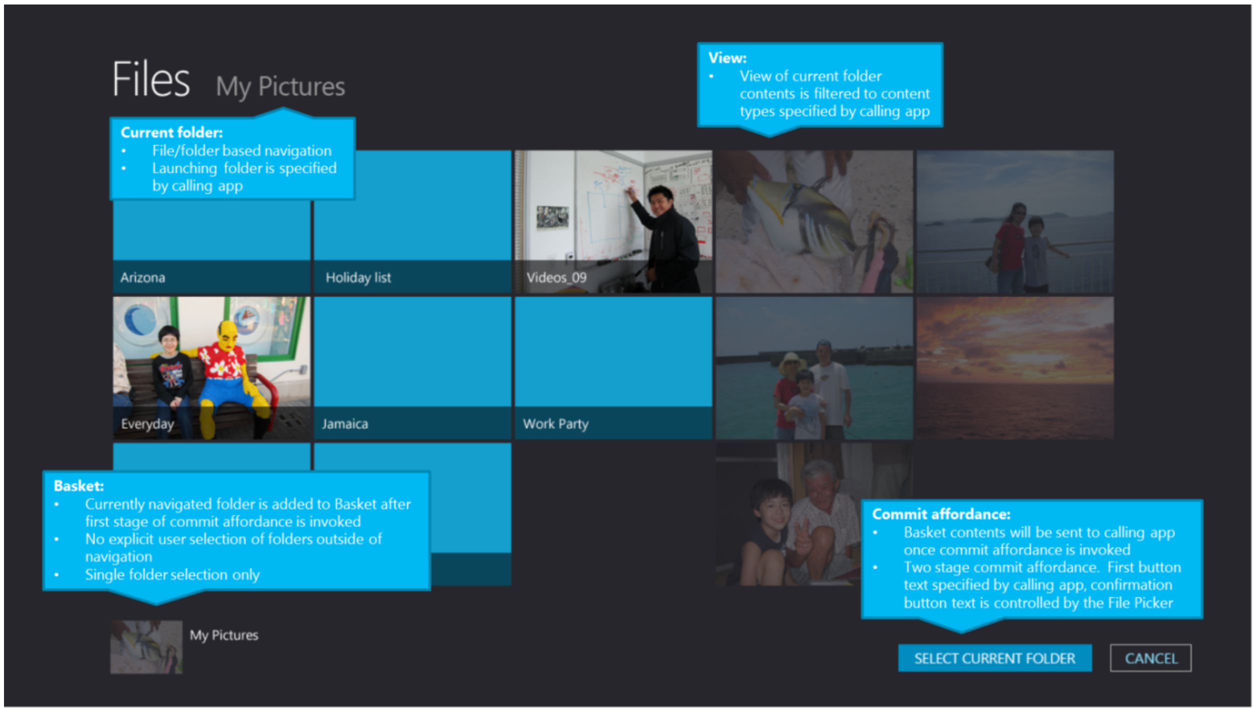
Current folder:
- File/folder based navigation
- Launching folder is specified by calling app
- Navigation is supported in Save mode
- File will be saved in currently navigated folder

File name:
- Default name specified by calling app
- Customizable by user
- One file can be saved at a time

Commit affordance:
- File will be saved to user specified location once commit affordance is invoked
- Button text specified by calling app

File type:
- Supported file types to save as (specified by calling app)
- User can choose desired type

Folder Mode:



View:
- View of current folder contents is filtered to content types specified by calling app

Current folder:
- File/folder based navigation
- Launching folder is specified by calling app

Basket:
- Currently navigated folder is added to Basket after first stage of commit affordance is invoked
- No explicit user selection of folders outside of navigation
- Single folder selection only

Commit affordance:
- Basket contents will be sent to calling app once commit affordance is invoked
- Two stage commit affordance. First button text specified by calling app, confirmation button text is controlled by the File Picker

4.13.1 THE FILE PICKER SYSTEM FLOW

To open an item:

1. The User launches a MoSh application.
2. The MoSh application launches the File Picker
3. The application creates an **IFileOpenPicker** object, calls the appropriate methods, and specifies the necessary properties on the object.
4. The user navigates and selects the desired files or folders. Note that the File Picker does not send callbacks to the application during picking.
5. When the user makes a selection, the associated **StorageFile** objects are handed back to the calling application.

To save an item:

1. The user launches a MoSh application.
2. The MoSh application launches the File Picker to save a file to a location
3. The application creates an **IFileSavePicker** object, calls the appropriate methods, and specifies the necessary properties on the object.
4. The user navigates and specifies the desired file name and type. Note that the File Picker does not send callbacks to the application while saving.
5. After the user confirms the save, the specified file is created in the specified location.

Three main interfaces expose the File Picker to the public: **IFilePicker**, **IFileOpenPicker**, and **IFileSavePicker**. **IFileOpenPicker** requires **IFilePicker**, and **IFileSavePicker** requires **IFilePicker**.

IFilePicker contains methods common in both open and save cases. **IFileOpenPicker** contains methods only applicable to open scenarios, and **IFileSavePicker** contains methods only applicable to save scenarios.

4.13.2 WINRT API DETAILS

IFileOpenPicker

Method	**Description**
PickSingleFileAsync	Displays the File Picker in single-select mode.
PickMultipleFilesAsync	Displays the File Picker in multi-select mode.
Property	**Description**
ViewMode	Determines the view mode of the File Picker (List or Thumbnail)
SettingsIdentifier	Assigns an identifier for a persisted state of the File Picker. Use this identifier to disambiguate between multiple File Picker instances in an application.
SuggestedStartLocation	Suggests a starting location (via **PickerLocationID**) if a File Picker location

	has not yet been persisted (via the **SettingsIdentifier** or the user already used the File Picker in that application).
CommitButtonText	Customizes the commit text in the File Picker.
FileTypeFilter	The File Picker filters the view contents and App-to-App picking locations based off of the extensions specified. **This is a required property.**

IFileSavePicker	
Method	**Description**
PickSaveFileAsync	Displays the File Picker in save mode.
Property	**Description**
SettingsIdentifier	Assigns an identifier for the persisted state to a File Picker. Use this identifier to disambiguate between multiple File Picker instances in an application.
SuggestedStartLocation	Suggests a starting location (via **PickerLocationID**) if a File Picker location has not yet been persisted (via the **SettingsIdentifier** or the user already used the File Picker in that application).
CommitButtonText	Customizes the commit text in the File Picker.
FileTypeChoices	Set of file types (extensions) that the calling application can save as.
DefaultFileExtension	Specifies a default file type that already exists in **FileTypeChoices**. If this property is not set, the default file type is the first file type specified in **FileTypeChoices.**
SuggestedSaveFile	Enables applications to take a **StorageFile** obtained through open or some other means and send it to the File Picker for a Save As… experience that selects the specified file. This property has priority over **SuggestedSaveFileName**.
SuggestedSaveFileName	Suggests a file name to the user.

IFolderPicker	
Method	**Description**
PickSingleFolderAsync	Displays the File Picker in folder mode.
Property	**Description**
ViewMode	Determines the view mode of the File Picker (List or Thumbnail)
SettingsIdentifier	Assigns an identifier for the persisted state to a File Picker. Use this identifier to disambiguate between multiple File Picker instances in an application.
SuggestedStartLocation	Suggests a starting location (via **PickerLocationID**) if a File Picker location has not yet been persisted (via the **SettingsIdentifier** or the user already

	used the File Picker in that application).
CommitButtonText	Customizes the commit text in the File Picker.
FileTypeFilter	The File Picker filters the view contents and App-to-App picking locations based off of the extensions specified. **This is a required property.**

4.13.3 SAMPLE JAVASCRIPT CODE

The following example creates a **FileOpenPicker**.

```
var openpicker = new Windows.Storage.FileOpenPicker();
openpicker.fileTypeFilter.replaceAll([".docx", ".doc"]);
openpicker.pickSingleFileAsync().then(function(file) {
    if (file) {
        DoSomething(file);
    }
});
```

The next example creates a **FileSavePicker**.

```
var savepicker = new Windows.Storage.FileSavePicker();
savepicker.fileTypeChoices.insert("JPEG File", [".jpeg", ".jpg"]);
savepicker.fileTypeChoices.insert("PNG File", [".png"]);
savepicker.pickSaveFileAsync().then(function (file) {
    if (file) {
        SaveFileTo(file);
    }
});
```

The last example creates a **FolderPicker**.

```
var picker = new Windows.Storage.FolderPicker();
picker.fileTypeFilter.replaceAll([".html", ".htm"]);
picker.pickSingleFolderAsync().then(function (folder) {
    if (folder) {
        DoSomethingWithFolder(folder);
    }
});
```

4.13.4 REFERENCES

1. File Picker Functional Spec
2. File Picker Dev Design
3. File Picker API Summary
4. File Picker Test Harness Instructions
5. MoSh App Data Model API Review

4.14 FILE PICKER EXTENSION

The optional File Picker extension category enables an application to make itself available in the File Picker (open mode only) as a hosted app where its contents are made available to the app that invoked the File Picker.

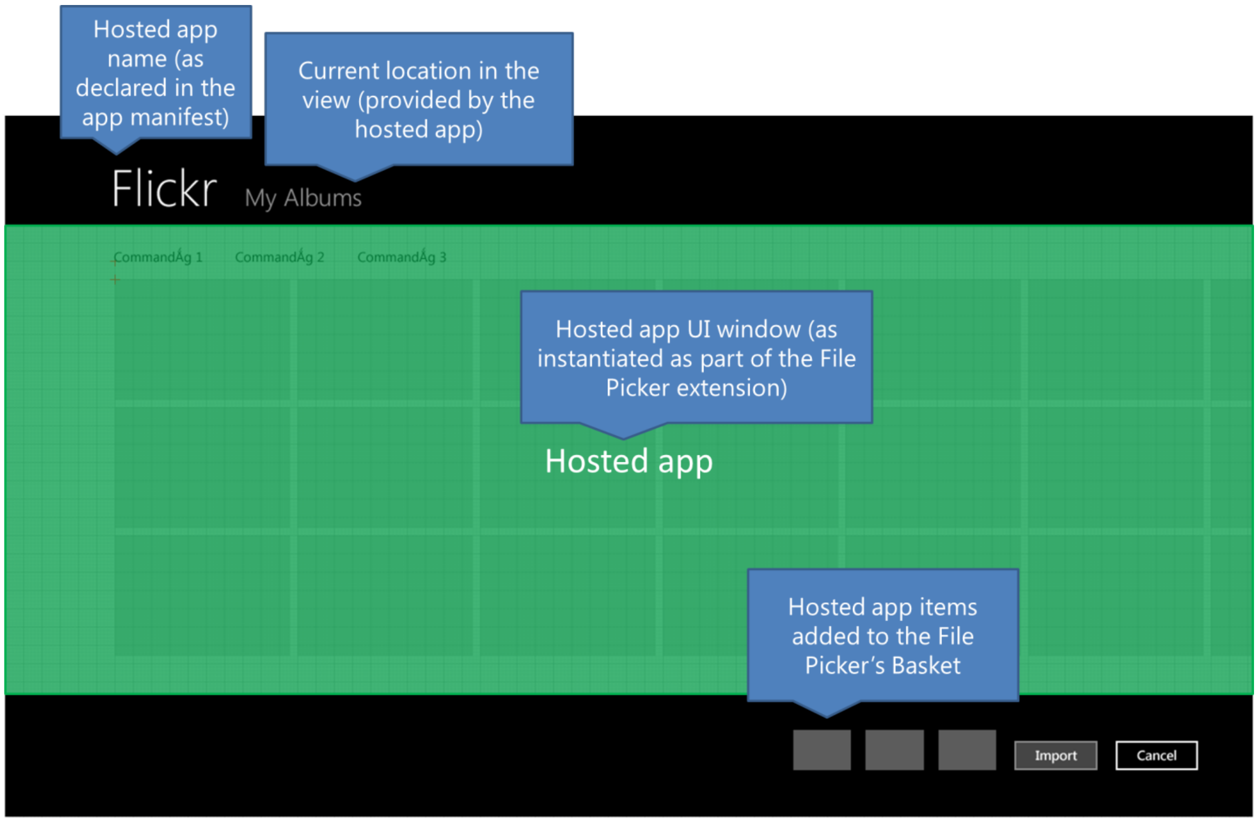

Hosted app name (as declared in the app manifest)
Current location in the view (provided by the hosted app)
Flickr
My Albums
CommandÁg 1
CommandÁg 2
CommandÁg 3
Hosted app UI window (as instantiated as part of the File Picker extension)
Hosted app
Hosted app items added to the File Picker's Basket
Import
Cancel

Figure 1: An application participating in the File Picker extension

File Picker extensions provide the following benefits:

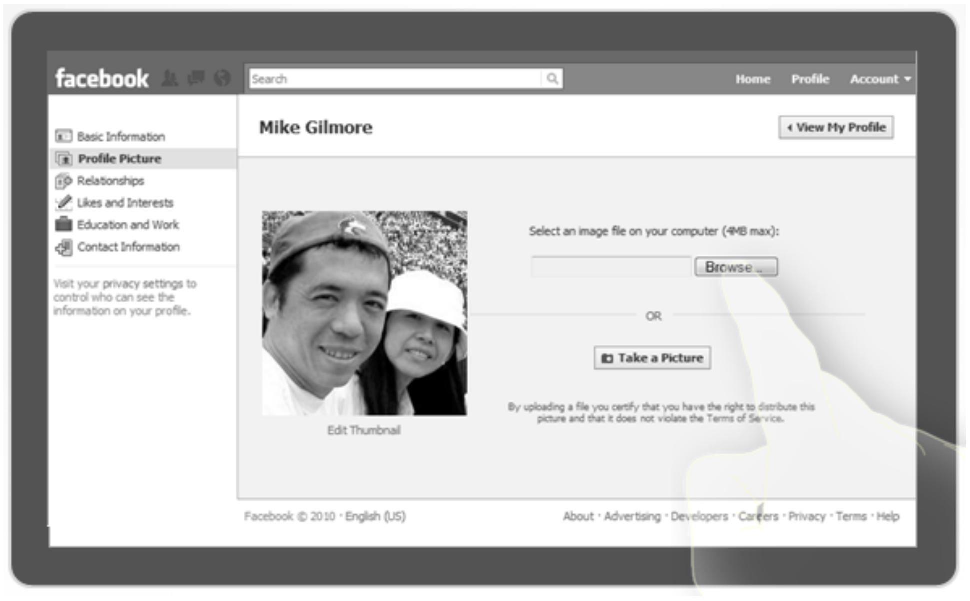
- Provides users with the freedom and flexibility to choose files that are stored and presented by applications.
- Provides a more accurate and complete picture of where data is stored.
- Provides a simple and consistent way to bring external items into an application.
- Enables applications to participate in additional user scenarios as a source of content.
- Can make an app more popular and encourage adoption.

The calling app is the application that originally invoked the File Picker and receives the picked items. This application is occluded while the File Picker is displayed and is largely uninvolved in the File Picker extension scenario, except as the ultimate endpoint of the data.

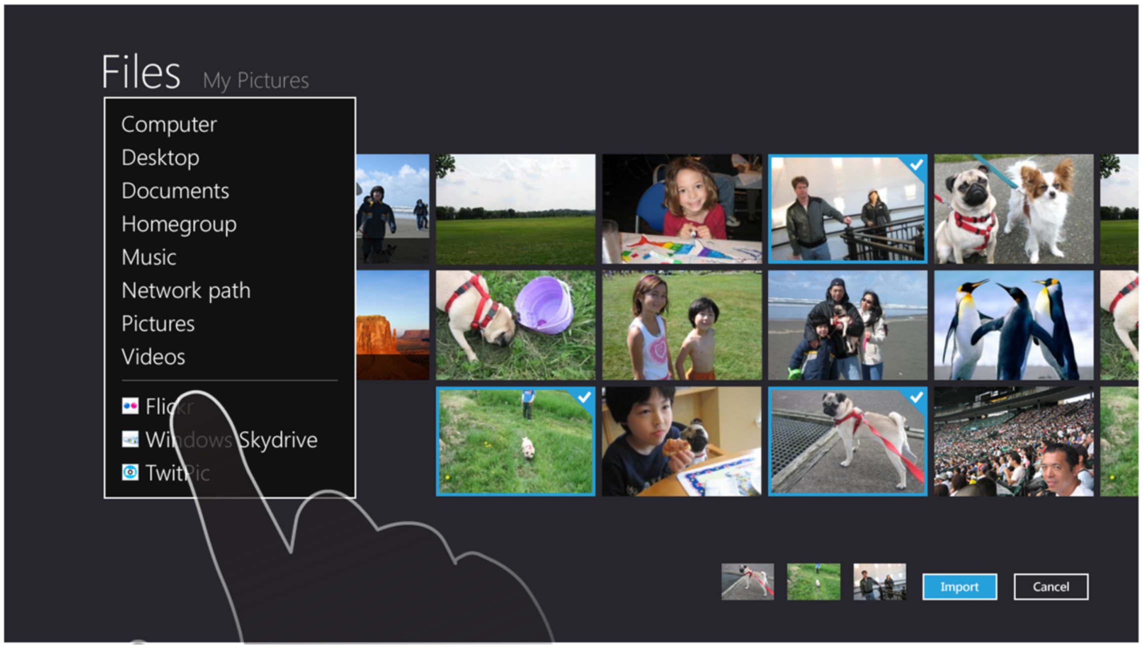
The hosted app is the application that has been selected by the user as a source of items to pick. This application is displayed non-modally inside the File Picker's UI, filling the "view" area of the File Picker. The File Picker interacts with this application programmatically to determine which items the user has selected. The File Picker extension APIs are only used by the hosted app to interact with the File Picker's Basket. The calling app uses the File Picker APIs (described in the preceding section) to retrieve items.

Apps declare supported file types (via file extensions) in their manifest. The calling app specifies which types of files it wants to open using the File Picker API. Hosted apps are filtered to show those that support at least one of the specified types. Apps may also specify that they support any type.

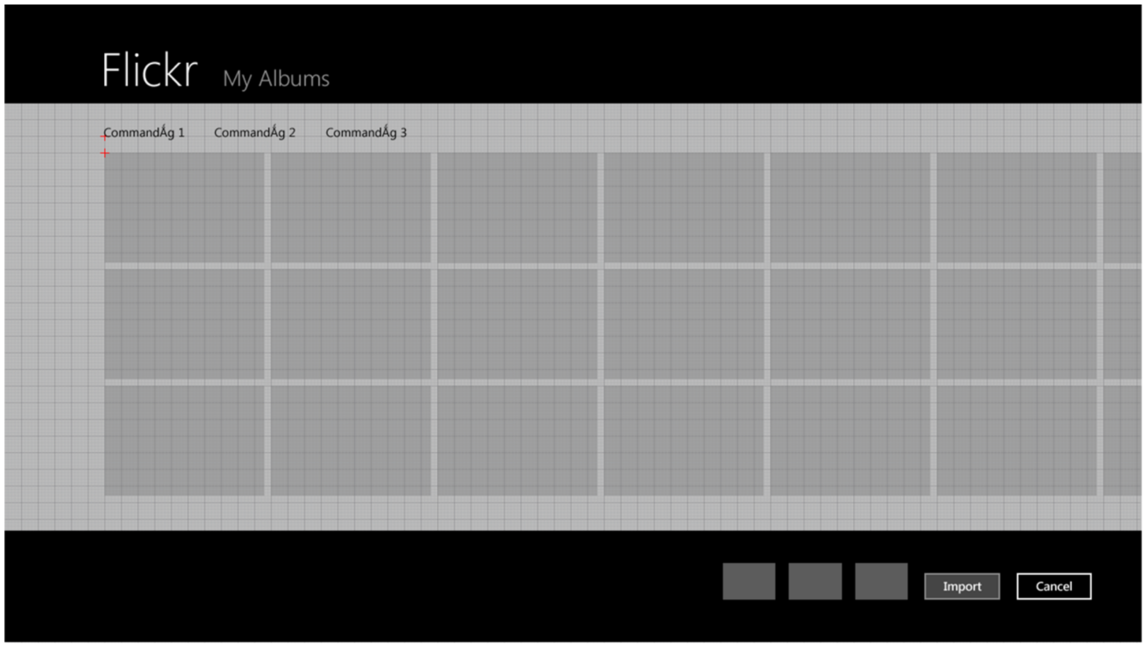
The following illustration shows a calling app invoking the File Picker through a browse for files button.


facebook
Search
Home
Profile
Account
Basic Information
Profile Picture
Relationships
Likes and Interests
Education and Work
Contact Information
Visit your privacy settings to control who can see the information on your profile.
Mike Gilmore
View My Profile
Select an image file on your computer (4MB max):
Browse...
OR
Take a Picture
By uploading a file you certify that you have the right to distribute this picture and that it does not violate the Terms of Service.
Edit Thumbnail
Facebook © 2010 · English (US)
About · Advertising · Developers · Careers · Privacy · Terms · Help

After picking some local items, user changes to a hosted app that's registered as a File Picker extension:



The hosted app's UI is shown and users can pick additional items:



4.14.1 HOSTED APP

The hosted app picking flow executes as follows**:**

1. When the user switches to a hosted app in the File Picker, a hosted window is created and the hosted app displays its UI within that window. A splash screen for the hosted app is also shown.

2. The hosted app is shown within the File Picker's window, filling up the view portion of the UI, but leaving the File Picker's Navigation and Basket areas available for user interaction. As the user selects items in the hosted app, the hosted app uses its item selection method to populate the File Picker's Basket and indicate what's been selected.
3. If the user removes an item from the File Picker's Basket, the hosted app receives an event it can use to update its UI to show that the item has been deselected.
4. The user commits the selection and the files that have been added to the File Picker's Basket by the hosted app are returned to the calling app.

4.14.2 WINRT API DETAILS

Windows.ApplicationModel.Activation.FilePickerActivatedEventArgs

Method	**Description**
Basket	This property provides access to the FilePickerBasket object that the application can use to interact programmatically with the FilePicker that it is hosted in to add/remove files from the Basket, etc.

Windows.Storage.FilePickerBasket

Method	**Description**
AddFile	When the user selects an item in the application's view, the application calls this method to add the item to the File Picker's Basket.
CanAddFile	An application hosted in the File Picker can determine whether a given file would be allowed in the Basket for the current File Picker session by calling this method. For example, if the calling app specified .jpg and .png as the file types to pick, a .bmp file would not be allowed in the Basket.
ContainsFile	This method enables an application hosted in the File Picker to determine whether a file is presently contained in the Basket.
RemoveFile	When the user deselects an item in the application's view, the application calls this method to remove the item to the File Picker's Basket.
AllowedFileTypes	Returns a list of file types (extensions) supported by the active instance of the File Picker. The value '*' anywhere in the list indicates that files of any type may be added to the Basket.
Title	Sets or retrieves the title displayed in the File Picker's UI to identify the current context of the hosted application.
Event	**Description**
FileRemoved	This event is raised by the File Picker when an item is removed from the Basket by the user.
FileRemovedEventHandler	Defines the delegate interface that an application should implement to receive FileRemoved events.

Windows.Storage.FileRemovedEventArgs

Property	Description
Id	This property contains the application-defined identifier of the file that was removed from the Basket. The value corresponds to the identifier provided by the application when the file was added to the Basket via FilePickerBasket.AddFile.

4.14.3 SAMPLE JAVASCRIPT CODE

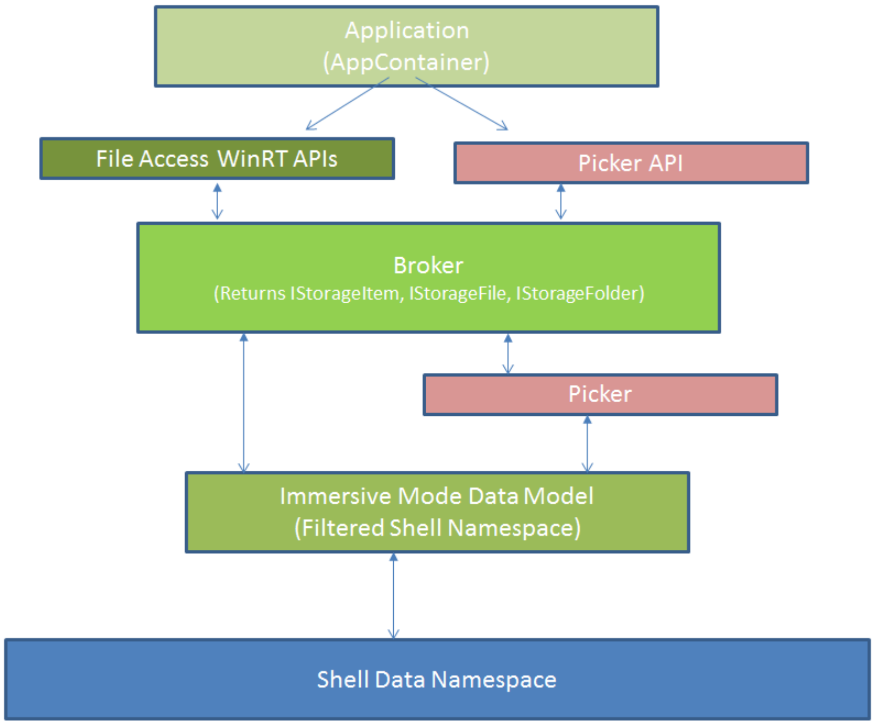
```
var basket;
var myItems; // populated with items by the application (app-defined
object type)
function activatedHandler(e) { // registered handler for
Windows.UI.WebUI.WebUIApplication.Activated event
    if (e.contractId === "Windows.FilePicker")
    {
        basket = e.basket;
        basket.addEventListener("fileremoved", function (sender, e) {
myItems[e.id].selected = false; });
        e.handled = true;
    }
}

function myOnItemTouched(item) {
    if (basket.addFile(item.id, item.file) ===
Windows.Storage.AddFileToBasketResult.AddedToBasket) { // item.file is a
Windows.Storage.StorageFile object
        item.selected = true;
    }
}
```

4.14.4 REFERENCES

1. App-to-App Picking Functional Spec
2. App-to-App Dev Design
3. App-to-App API Spec
4. App-to-App API Summary

4.15 FILE SYSTEM BROKER

This contract enables MoSh apps to access user content by picking files or folders through the MoSh Item Picker or by programmatically selecting files or folders. All programmatic access APIs are brokered through the File Broker to provide safe programmatic access for Application Container apps.


Application
(AppContainer)
File Access WinRT APIs
Picker API
Broker
(Returns IStorageItem, IStorageFile, IStorageFolder)
Picker
Immersive Mode Data Model
(Filtered Shell Namespace)
Shell Data Namespace

The file broker is a medium integrity level process that acts as an intermediary between an Application Container app and user content. It intercepts calls for file access to returns **IStorageItem** objects back to the app.

By default, apps to not have access to files or folders; they must declare capabilities to gain access. Apps may request access to the following folders by declaring capabilities in the package manifest:

- Music Library (Local and HomeGroup)
- Pictures Library (Local and HomeGroup)
- Videos Library (Local and HomeGroup)
- Recorded TV Library (Local and HomeGroup)
- Documents Library (Local and HomeGroup)
    - Must also specify supported file types
- The HomeGroup root node
- Removable Devices

Applications need to specify file types for the Documents Library and Removable Devices locations in the manifest. Users can grant access to additional locations by using the Picker, even if that app doesn't have file access capabilities.

4.15.1 THE FILE ACCESS SYSTEM FLOW

1. The app declares file access capabilities in its manifest.

2. When the app runs, SIDs, Security Identifiers, that map to the capabilities (Capability SIDs) and a Package SID are applied to the process.
3. Apps request access to files within a location using the WinRT API.
4. The file broker intercepts the call and invokes the **AccessCheck** method.
5. The **AccessCheck** method uses SID values to determine whether the app has access.
6. If the file broker access check is successful, the file broker returns an **IStorageItem (IStorageFile/IStorageFolder)** to the app.

4.15.2 WINRT API DETAILS

StorageFolder
Covers APIs for a storage folder (or StorageFolder) storage item object, which can also be referred to as a storage folder namespace. The storage folder namespace is "Windows.Storage.StorageFolder".

Method	**Description**
createFileAsync()	Creates a file within the current StorageFolder storage item object.
createFolderAsync()	Create a folder within the current StorageFolder storage item object.
getIndexedStateAsync()	Function to determine if the current StorageFolder storage item object represents a location which has its items' properties stored in a database for faster property retrieval.
areQueryOptions Supported (QueryOptions)	Checks, and returns an indication of whether, the passed query options can be applied to the current StorageFolder storage item object.
isCommonFileQuery Supported (CommonFileQuery)	Checks, and returns an indication of whether, the passed common file query options (one or more options pre-defined for application use) can be applied to the current StorageFolder storage item object.
isCommonFolder QuerySupported (CommonFolderQuery)	Checks, and returns an indication of whether, the passed common folder query options (one or more options pre-defined for application use) can be applied to the current StorageFolder storage item object.
getFileAsync(name)	Function to perform a single file retrieval.
getFolderAsync(name)	Function to perform a single folder retrieval.
getItemAsync(name)	Function to perform a single file or folder retrieval.

StorageFile
Illustrates APIs for a storage file (or StorageFile) storage item object. The storage file namespace is "Windows.Storage.StorageFile".

Method	**Description**
fileName	The name of the file represented by the current StorageFile storage item object.
fileType	The item type (e.g., picture, video, document) of the file represented by the current StorageFile storage item object.
openAsync	Opens a random access stream for consumption, allowing the data for the file represented by the current StorageFile storage item object to be provided to the isolated

	application for both read and write.
openForReadAsync	Opens an input stream allowing the isolated application to read data for the file represented by the current StorageFile storage item object.
copyAsync (destinationFolder, newFileName, nameCollisionOption)	Copy a single file to a new location specified by destinationFolder. The name of the new file can optionally be specified as newFileName, and collision options (e.g., how to resolve collisions/duplicates in file names) can optionally be specified as nameCollisionOption.
copyAndReplaceAsync(fileToReplace)	Copy a single file represented by the current StorageFile storage item object to a specified location, and overwrite the file at that specified location.
moveAsync (destinationFolder, newFileName, nameCollisionOption)	Move a single file to a new location specified by destinationFolder. The name of the new file can optionally be specified as newFileName, and collision options (e.g., how to resolve collisions/duplicates in file names) can optionally be specified as nameCollisionOption.
moveAndReplaceAsync(fileToReplace)	Move a single file represented by the current StorageFile storage item object to a specified location, and overwrite the file at that specified location.

Known Folders
Illustrates APIs for a known folders namespace, which refers to a particular set of pre-defined set of folders or libraries of a file system item source that can be accessed by AppContainers. The known folders namespace is "Windows.Storage.KnownFolders".

Method	Description
musicLibrary	Returns a storage item object representing a Music Library folder, including identification of files stored in the Music Library folder.
picturesLibrary	Returns a storage item object representing a Pictures Library folder, including identification of files stored in the Pictures Library folder.
videosLibrary	Returns a storage item object representing a Videos Library folder, including identification of files stored in the Videos Library folder.
recordedTVLibrary	Returns a storage item object representing a Recorded TV Library folder, including identification of files stored in the Recorded TV Library folder.
documentsLibrary	Returns a storage item object representing a Documents Library folder, including identification of files stored in the Documents Library folder.
homeGroup	Returns a storage item object representing a Homegroup folder, including identification of files stored in the Homegroup folder.
removableDevices	Returns a storage item object representing a Removable Devices folder, including identification of files stored in the Removable Devices folder.
mediaServerDevices	Returns a storage item object representing a Media Server Devices

	folder, including identification of files stored in the Media Server folder.

Retrieve particular files/folders
Illustrates APIs to store or retrieve particular files or folders. The storage namespace is "Windows.Storage".

Method	Description
DownloadsFolder	Returns a storage item object representing a Downloads folder, including identification of files stored in the Downloads folder.
getFileFromPathAsync(path)	Returns a StorageFile storage item object from the location specified by the given path.
getFileFromUriAsync(uri)	Returns a StorageFile storage item object from the location specified by the given URI.
createFileForTransferAsync()	Returns a storage item object that is a Transfer File (a temporary StorageFile storage item object to which streamed data can be subsequently written).
getFolderFromPathAsync()	Returns a StorageFolder storage item object from the location specified by the given path

Operations
Illustrates APIs for a storage item object, which can also be referred to as a storage item namespace. The storage item namespace is "Windows.Storage.StorageItem".

Method	Description
getThumbnailAsync()	Function to get the thumbnail of the item represented by the current storage item object.
renameAsync()	Function to change the name of the item represented by the current storage item object to a name specified by the isolated application.
deleteAsync()	Function to delete the item represented by the current storage item object.
isOfType()	Function to check whether the current storage item object is a StorageFile storage item object or StorageFolder storage item object.
name	The name of the item represented by the current storage item object.
size	The size (e.g., in bytes) of the item represented by the current storage item object.
displayType	The item type (e.g., picture, video, document) of the item represented by the current storage item object, formatted for consumption by the user.
contentType	The item type (e.g., picture, video, document) of the item represented by the current storage item object.

path	A path identifying where the item represented by the current storage item object is located.
attributes	Attributes of the item represented by the current storage item object maintained by the item source.
dateModified	A date that the item represented by the current storage item object was most recently modified.
dateCreated	A date that the item represented by the current storage item object was created.
folderRelativeId	An identifier that can be used to uniquely identify an item relative to its parent folder.
ExtraProperties	Additional "System" properties of the item represented by the current storage item object that are maintained by the item source and can be retrieved.
getMusicPropertiesAsync()	Function to retrieve music-specific properties of an item of a music item type that are accessible via an asynchronous (async) call.
getVideoPropertiesAsync()	Function to retrieve video-specific properties of an item of a music item type that are accessible via an asynchronous call.
getImagePropertiesAsync()	Function to retrieve picture- or image-specific properties of an item of a music item type that are accessible via an asynchronous call.
getDocumentPropertiesAsync()	Function to retrieve document-specific properties of an item of a music item type that are accessible via an asynchronous call.

Query options
Illustrates APIs for query options to specify query options for search requests. The namespace is "Windows.Storage.QueryOptions".

Method	**Description**
fileTypeFilter	Adds a file extension filter to the query to specify particular file extension types to be searched for.
folderDepth	Specifies whether the query is deep or shallow (e.g., whether sub-folders are to be searched)
applicationSearchFilter	Adds an AQS search filter to the query.
userSearchFilter	Adds a second AQS search filter to the query.
searchLocale	Specifies the search locale (e.g., item source or location of an item source) of the query.
indexerOption	Specifies whether only locations that are "indexed" (have their item properties cached in a database) should be queried.
sortOrder	Adds an "OrderBy" sort to the query, specifying a particular order for arranging the storage item objects returned by the search.
stackPropertyName	Adds a "GroupBy" shape to the query, specifying a particular grouping for arranging the storage item objects returned by the search.

dateStackOption	For items grouped by date, indicates if the grouping should be by day, month, year, etc.
saveToString()	Function to save the query options to a string to handle tombstoning (e.g., forced suspension of the isolated application, such as to reduce power consumption).
loadFromString(string)	Function to load the query options from a string to handle tombstoning (e.g., forced suspension of the isolated application, such as to reduce power consumption).

Query

Illustrates APIs for a set of interfaces allowing isolated applications to submit queries or searches for items. The query namespace is "Windows.Storage".

Method	**Description**
StorageQueryResultBase. contentsChanged	Event that is fired when the file item contents behind a query have changed.
StorageQueryResultBase. optionsChanged	Event that is fired when the query options for a query have been changed.
StorageQueryResultBase. findStartIndexAsync()	Allows the app to look up the index of the first item that corresponds to the provided value of the first sort order property.
StorageQueryResultBase. getCurrentQueryOptions()	Function to retrieve the current query options of the current query.
StorageQueryResultBase. applyNewQueryOptions(queryOptions)	Function to apply new query options specified as queryOptions to the current query.
StorageQueryResultBase. getItemCountAsync()	Function to get the number of items behind (satisfied by) a query.
StorageFileQueryResult. getFilesAsync()	Function to retrieve files satisfied by a query.
StorageFileQueryResult. getFilesAsync(start, count)	Function to retrieve from a query a number of files specified by count and beginning at an index specified by start.
StorageFolderQueryResult. getFoldersAsync()	Function to retrieve folders satisfied by a query.
StorageFolderQueryResult. getFoldersAsync(start, count)	Function to retrieve from a query a number of folders specified by count and beginning at an index specified by start.
StorageItemQueryResult. getItemsAsync()	Function to retrieve items satisfied by a query
StorageItemQueryResult. getItemsAsync(start, count)	Function to retrieve from a query a number of items specified by count and beginning at an index specified by start.

Fast Accessors
Illustrates APIs for fast accessors hich is a set of interfaces allowing items to be retrieved (e.g., quickly, without specifying an AQS query). The APIs can support retrieving items in different ways, such as a shallow mode (e.g., returning results from a particular folder or directory), a deep mode (e.g., returning results from a particular folder or directory as well as all sub-folders or sub-directories), and so forth. The quick accessors namespace is "Windows.Storage.StorageFolder".

Method	Description
getFilesAsync()	Retrieve files arranged in a by-folder grouping.
getFiles(CommonFileQuery)	Retrieve files using the passed common file query options.
getFiles(CommonFileQuery, start, count)	Retrieve, using the passed common file query options, a number of files specified by count and beginning at an index specified by start.
getFoldersAsync()	Retrieve folders arranged in a by-folder grouping.
getFoldersAsync(CommonFolderQuery)	Retrieve folders using the passed common folder query options.
getFoldersAsync(CommonFolderQuery, start, count)	Retrieve, using the passed common folder query options, a number of folders specified by count and beginning at an index specified by start.
getItemsAsync()	Retrieve files and folders arranged in a by-folder grouping.
getItemsAsync(start, count)	Retrieve, arranged in a by-folder grouping, a number of files and folders specified by count and beginning at an index specified by start.

Queries
Illustrates APIs for a query creation namespace, which is a set of interfaces allowing queries to be created by isolated applications. Once created, these queries can be maintained by the broker module and subsequently accessed by the appContainers creating the query. The query creation namespace is "Windows.Storage.StorageFolder".

Method	Description
createFileQuery()	Creates a query with files arranged in a by-folder grouping.
createFileQuery(CommonFileQuery)	Creates a file-only query using the passed common file query options.
createFileQueryWithOptions(QueryOptions)	Creates a file-only query with query options specified by the isolated application.
createFolderQuery()	Creates a query with folders arranged in a by-folder grouping.
createFolderQuery(CommonFolderQuery)	Creates a folder-only query using the passed common folder query options.
createFolderQueryWith Options(QueryOptions)	Creates a folder-only query with query options specified by the isolated application.
createItemQuery()	Creates a query with files and folders arranged in a by-folder grouping.
createItemQueryWithOptions(	Creates a file and folder query with query options specified by the

QueryOptions)	isolated application

Storage Item Persistance
Illustrates APIs for a set of interfaces allowing AppContainers to persist storage items. Storage item objects can be persisted across multiple executions of an isolated application. The storage item persistence namespace is "StorageApplicationPermissions.futureAccessList".

Method	Description
add(storageItem, metadata)	Adds the specified storage item object to the persisted access list, and returns a token to the isolated application allowing the persisted storage item object to be subsequently retrieved. The isolated application can optionally specify, as metadata, metadata to be linked to the persisted storage item object.
addOrReplace(token, storageItem)	Adds the specified storage item object to the persisted access list as with add(), but also provides the ability to replace any existing persisted access list entry corresponding to the token. The isolated application can optionally specify, as metadata, metadata to be linked to the persisted storage item object.
getItemAsync(token)	Get the persisted storage item object specified by the token from the persisted access list.
getFileAsync(token)	Get the persisted StorageFile storage item object specified by the token from the persisted access list.
getFolderAsync(token)	Get the persisted StorageFolder storage item object specified by the token from the persisted access list.
remove(token)	Remove the persisted storage item object specified by the token from the persisted access list.
containsItem(token)	Check, and return an indication of, whether the storage item object specified by the token is present in the persisted access list.
clear()	Clears the access list, removing all persisted storage item objects from the persisted access list.
checkAccess(storageItem)	Check, and return an indication of, whether the isolated application has access to the specified storage item object (e.g., through a capability or because the item or parent is persisted).
entries	Retrieves the entire set of persisted storage item objects in the persisted access list.
maximumItemsAllowed	Retrieves the maximum number of storage item objects that are permitted to be persisted in the persisted access list.

Most Recently Used (MRU)
Illustrates APIs for a most recently used (MRU) list, which is a set of interfaces allowing AppContainers to generate and maintain a list of most recently used items. The MRU list is an example of persisted storage item objects, with the persisted access list being the MRU list. The MRU list namespace is "StorageApplicationPermissions.mostRecentlyUsedList".

Method	Description
add(storageItem, metadata)	Adds the specified storage item object to the MRU list, and returns

	a token to the isolated application allowing the persisted storage item object to be subsequently retrieved. The isolated application can optionally specify, as metadata, metadata to be linked to the persisted storage item object.
addOrReplace(token, storageItem)	Adds the specified storage item object to the MRU list as with add(), but also provides the ability to replace any existing MRU list entry corresponding to the token. The isolated application can optionally specify, as metadata, metadata to be linked to the persisted storage item object.
getItemAsync(token)	Get the persisted storage item object specified by the token from the MRU list.
getFileAsync(token)	Get the persisted StorageFile storage item object specified by the token from the MRU list.
getFolderAsync(token)	Get the persisted StorageFolder storage item object specified by the token from the MRU list.
remove(token)	Remove the persisted storage item object specified by the token from the MRU list.
containsItem(token)	Check, and return an indication of, whether the storage item object specified by the token is present in the MRU list.
clear()	Clears the MRU list, removing all persisted storage item objects from the MRU list.
checkAccess(storageItem)	Check, and return an indication of, whether the isolated application has access to the specified storage item object (e.g., through a capability or because the item or parent is persisted).
entries	Retrieves the entire set of persisted storage item objects in the MRU list.
maximumItemsAllowed	Retrieves the maximum number of storage item objects that are permitted to be persisted in the MRU list.

4.15.3 SAMPLE JAVASCRIPT CODE

This example shows how to access and read all the files in the Music Library sorted by Atrist, Album, Track and walking the list:

```
var library = Windows.Storage.KnownFolders.musicLibrary;
library.getFilesAsync(Windows.Storage.CommonFileQuery.orderByMusicInfo).th
en(function(files) {
    files.forEach(function(file) {
        doSomething(file);
    });
});
```

The next example shows how to search and enumerate the results.

```
var options = new
Windows.Storage.QueryOptions(Windows.Storage.CommonFileQuery.orderBySearch
Rank);
options.applicationSearchFilter = "(wedding OR reception) AND kind:photo";
options.searchLocale = "en-us";
var queryResult =
Windows.Storage.KnownFolders.picturesLibrary.createItemQueryWithOptions(op
tions);
queryResult.getItemsAsync().then(function(items) {
    items.forEach(function(item) {
        if (item.isOfType(Windows.Storage.StorageItemTypes.folder)) {
            myAddItemInFolderToSlideshow(item);
        } else if (item.isOfType(Windows.Storage.StorageItemTypes.file)) {
            myAddItemToSlideshow(item);
        }
    });
});
```

This code example enumerates a photo collection and displays the photos' thumbnails and filenames, organized in a year/month/list hierarchy:

```
var library = Windows.Storage.KnownFolders.picturesLibrary;

library.getFoldersAsync(Windows.Storage.CommonFolderQuery.groupByYear).the
n(function(yearList) {
    yearList.forEach(function(yearFolder) {
       var yearName = yearFolder.name;


yearFolder.getFoldersAsync(Windows.Storage.CommonFolderQuery.groupByMonth)
.then(function(monthList) {

           monthList.forEach(function(monthFolder) {
                var monthName = monthFolder.name;

                monthFolder.getFilesAsync().then(function(files) {
                    files.forEach(function(file) {
                         file.getThumbnailAsync().then(function(thumbnail)
{
                             // Do something with thumbnail...
                         });
                    });
                });
           });
        });
    });
});
```

4.15.4 REFERENCES

1. File Broker API Doc
2. File Broker Functional Spec
3. File Broker Dev Design

4.16 CONNECT – PRINT

The Connect Charm enables the user to extend the modern Windows experience through devices. The Connect Charm is a user experience hosted by the Charms Bar in the Edgy window. The Connect Charm is invoked by activating the Charms Bar through interaction with Edgy, then selecting the Connect Charm from the Charms Bar. Selection launches a flyout that provides quick access to device-related activities.

The Print application contract enables developers to use the Print button on the Connect Charm to print. In response to the button click, the developer can invoke the Windows print experience or provide a custom experience. This contract only has a source application component.

The source application is the active MoBody application; it is this application that responds to the Print button click.

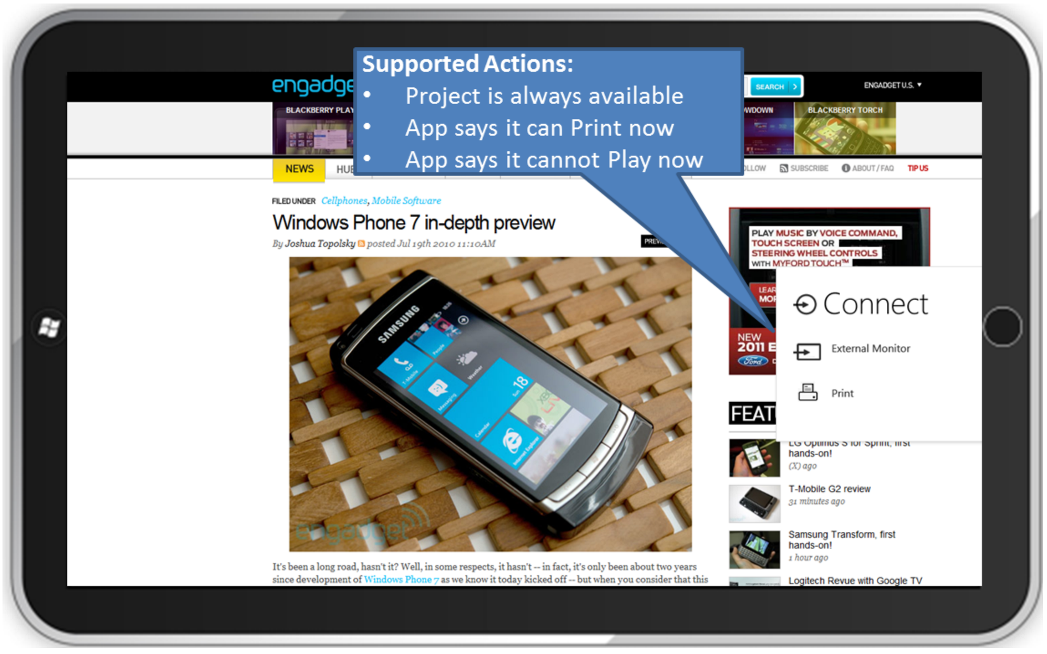
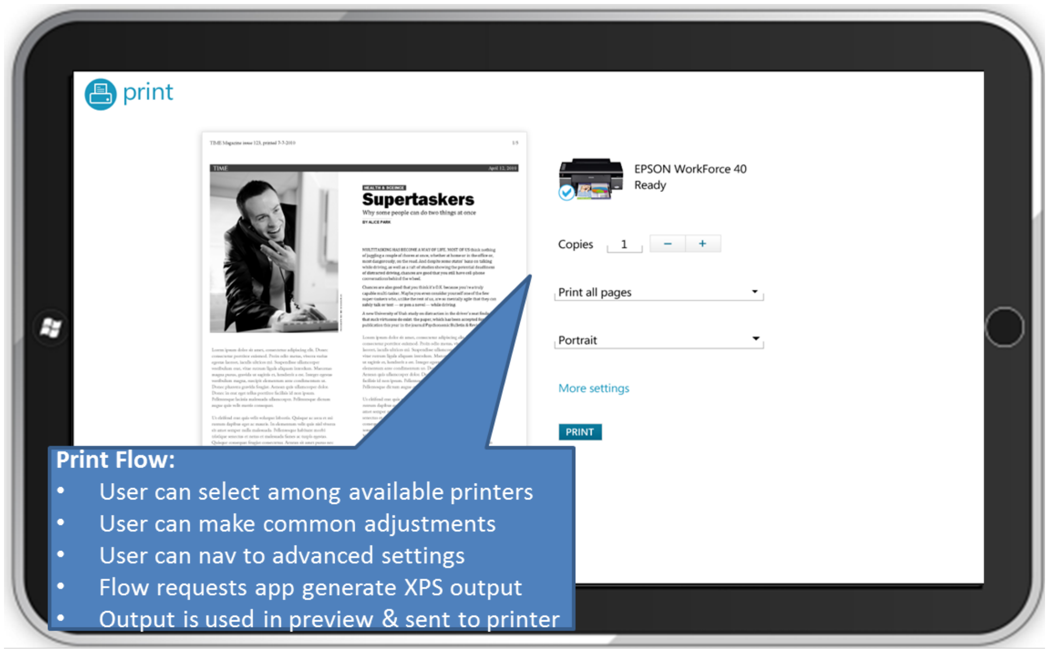

Supported Actions:
• Project is always available
• App says it can Print now
• App says it cannot Play now
Connect
External Monitor
Print
Windows Phone 7 in-depth preview

Figure 4: The Connect Charm



Print Flow:

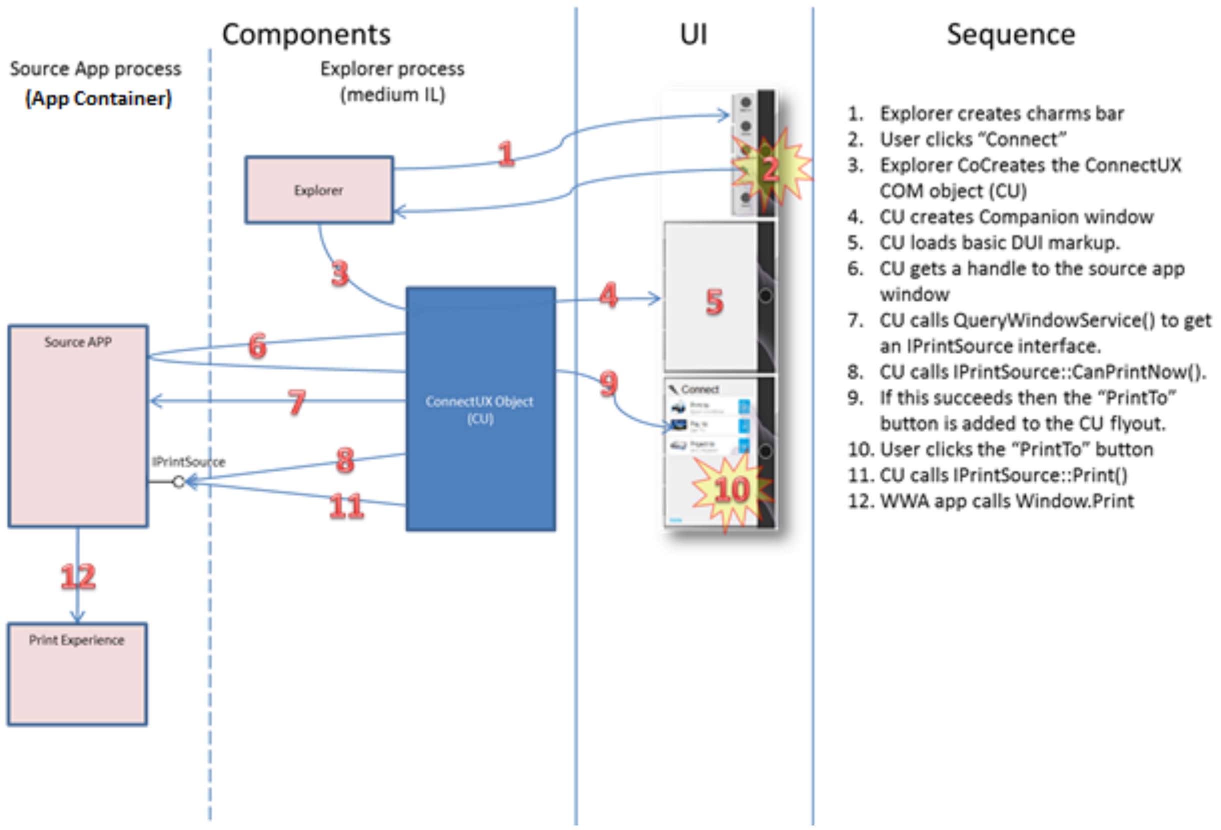
- User can select among available printers
- User can make common adjustments
- User can nav to advanced settings
- Flow requests app generate XPS output
- Output is used in preview & sent to printer

Figure 5: An app participating in the Print contract

4.16.1 THE PRINT SOURCE APP CONTRACT

The Print contract is optional in several ways. A source app may decline to opt-in to the Print button on the Connect Charm, in which case the Connect Charm will never show the Print button for that app. In such a case, the app may nonetheless support printing, even using the Windows Print experience, because printing is an activity that an app can perform entirely within its own process. If the source app opts-in to the Print button on the Connect Charm, the app may inform the Connect Charm on demand that it is not able to Print at the moment, in which case the Print button will not be displayed on the Connect Charm.

The Print flow executes as shown in the following illustration.



1. Explorer creates charms bar
2. User clicks "Connect"
3. Explorer CoCreates the ConnectUX COM object (CU)
4. CU creates Companion window
5. CU loads basic DUI markup.
6. CU gets a handle to the source app window
7. CU calls QueryWindowService() to get an IPrintSource interface.
8. CU calls IPrintSource::CanPrintNow().
9. If this succeeds then the "PrintTo" button is added to the CU flyout.
10. User clicks the "PrintTo" button
11. CU calls IPrintSource::Print()
12. WWA app calls Window.Print

Figure 6: The Print flow

4.16.2 SAMPLE JAVASCRIPT CODE

Note that the print contract isn't available in PDC-4. However, the **Window.Print** method is available.

4.16.3 REFERENCES

1. Print Dev Spec

4.17 CONNECT - PLAYTO

The PlayTo contract enables developers to use the PlayTo button on the Connect Charm to invoke remote-play behavior. In response to the button click, the application can invoke the Windows PlayTo experience or provide a custom PlayTo experience. This contract only has a source application component.

The source application is the active MoBody application; it is this application that may respond to the PlayTo button click on the Connect Charm.

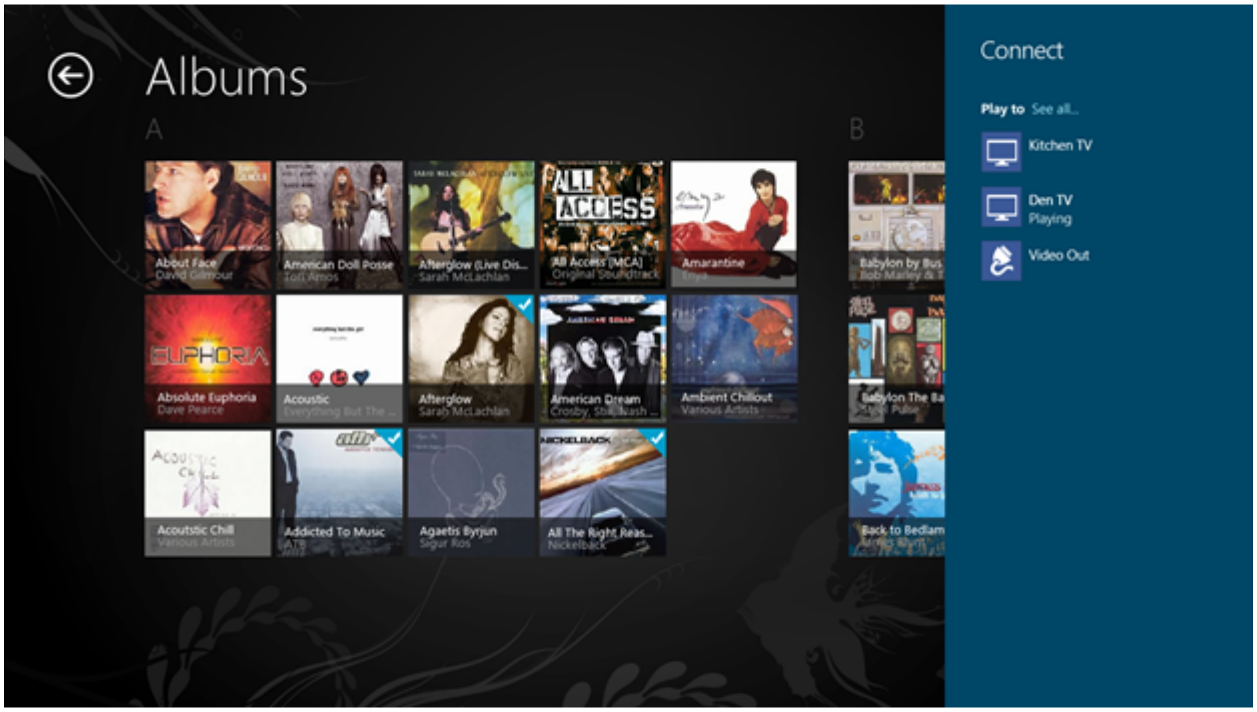

Albums
A
B
Connect
Play to See all...
Kitchen TV
Den TV
Playing
Video Out
About Face David Gilmour
American Doll Posse Tori Amos
Afterglow (Live Dis... Sarah McLachlan
All Access [MCA] Original Soundtrack
Amarantine Enya
Babylon by Bus Bob Marley & T
Absolute Euphoria Dave Pearce
Acoustic Everything But The ...
Afterglow Sarah McLachlan
American Dream Crosby, Still, Nash ...
Ambient Chillout Various Artists
Babylon The Ba Steel Pulse
Acoustic Chill Various Artists
Addicted To Music ATB
Agaetis Byrjun Sigur Ros
All The Right Reas... Nickelback
Back to Bedlam James Blunt

Figure 9: The Connect Charm showing the PlayTo button

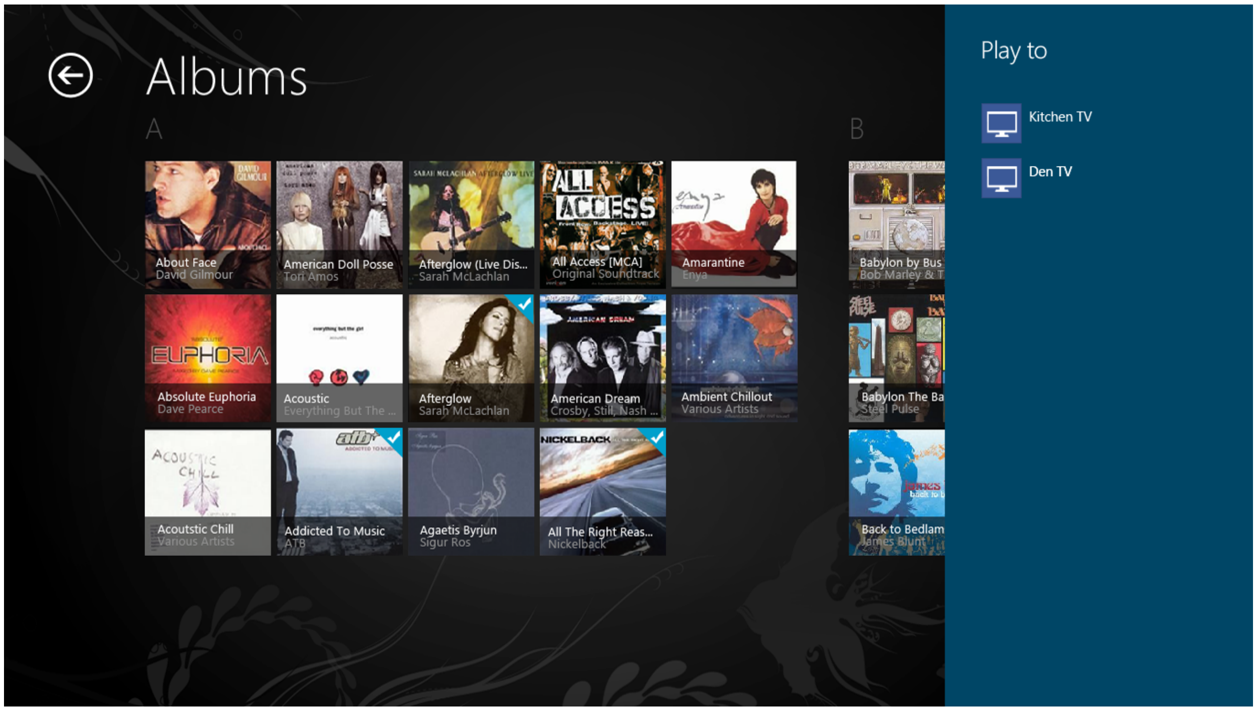

Albums
A
B
Play to
Kitchen TV
Den TV
About Face David Gilmour
American Doll Posse Tori Amos
Afterglow (Live Dis... Sarah McLachlan
All Access [MCA] Original Soundtrack
Amarantine Enya
Babylon by Bus Bob Marley & T
Absolute Euphoria Dave Pearce
Acoustic Everything But The ...
Afterglow Sarah McLachlan
American Dream Crosby, Still, Nash ...
Ambient Chillout Various Artists
Babylon The Ba Steel Pulse
Acoustic Chill Various Artists
Addicted To Music ATB
Agaetis Byrjun Sigur Ros
All The Right Reas... Nickelback
Back to Bedlam James Blunt

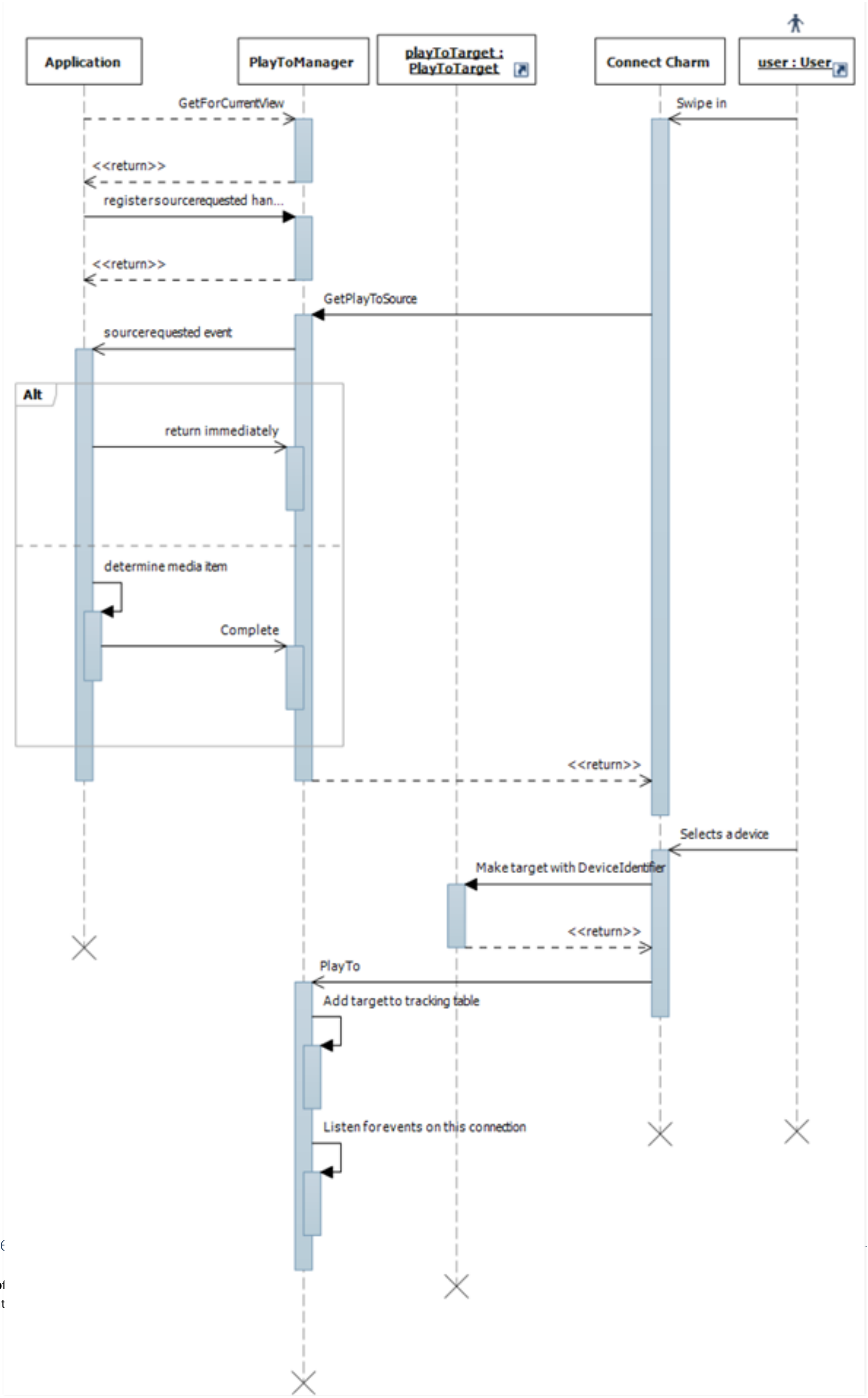
Figure 10: The PlayTo device picker

4.17.1 THE PLAYTO SOURCE APP CONTRACT

The PlayTo contract enables developers to use the PlayTo button on the Connect Charm to invoke remote-play behavior. In response to the button click, the application can choose to participate in the Windows PlayTo experience. This contract only has a source application component.

The source application is the active MoBody application; it is this application that may respond to the PlayTo button click on the Connect Charm.

The PlayTo flow executes as shown the following illustration**:**



Application
PlayToManager
playToTarget : PlayToTarget
Connect Charm
user : User
GetForCurrentView
Swipe in
<<return>>
registersourcerequested han...
<<return>>
GetPlayToSource
sourcerequested event
Alt
return immediately
determine media item
Complete
<<return>>
Selects a device
Make target with DeviceIdentifier
<<return>>
PlayTo
Add target to tracking table
Listen for events on this connection

Figure 7: The PlayTo flow

4.17.2 WINRT API DETAILS

PlayToManager Object

Property	**Description**
None	
Events	**Description**
SourceRequested	Raised when the user has started opening the Connect Charm.
Method	**Description**
ShowPlayToUI	Asks the PlayToManager to open the Connect Charm for the user.
GetForCurrentView	Returns a valid instance

PlayToSourceRequest Object

Property	**Description**
Deadline	Time in which the application must supply a source
Events	**Description**
None	
Method	**Description**
DisplayErrorString	
GetDeferral	Called when the application needs to perform asynchronous operations to retrieve the current PlayTo source.
SetSource	Called by the application to provide a PlayTo source to the manager.

PlayToSourceDeferral Object

Property	**Description**
None	
Events	**Description**
None	
Method	**Description**
Complete	Indicates the application has finished preparing a source object. This unblocks the PlayToManager, allowing the Connect Charm to display devices that are compatible with the application's content.

4.17.3 SAMPLE JAVASCRIPT CODE

This demonstrates the PlayTo fucnctionality:

```
function initialize() {
    var pm = Windows.UI.Immersive.PlayTo.PlayToManager.getForCurrentView();
    pm.addEventListener("sourcerequested", sourceRequestedHandler, false);
}
```

```
function sourceRequestedHandler(e) {
   var sr = e.sourceRequest;

   var videoElement = document.getElementById('videoplayer')
   var controller = videoElement.msPlayToSource;

   // select the video element to use for PlayTo
   sr.setSource(controller);
}

document.addEventListener("DOMContentLoaded", initialize, false);
```

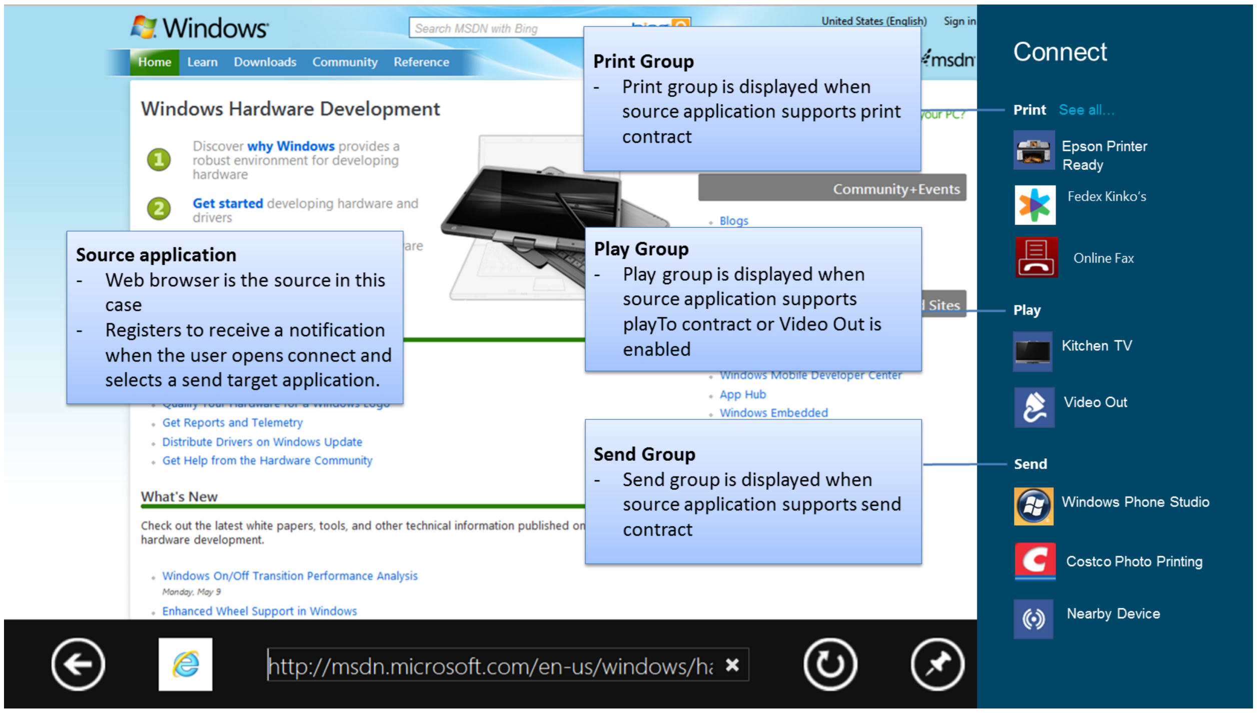
4.17.4 REFERENCES

1. PlayTo Functional Spec

4.18 CONNECT – SEND

Send enables users to share content from their current application with another device application. Developers of any Windows application can utilize the Windows Send functionality to enable their devices to send content. Apps that provide send-able content are called source applications. The source application is the active application in the MoBody. An example source application is a magazine reader.

Developers of send applications can utilize the send functionality so that users can send to their application from any source application. Apps that receive send-able content and make it visible to devices are called target applications. The target application is displayed in a flyout window. An example target application is Windows Phone Application.


Print Group
- Print group is displayed when source application supports print contract

Source application
- Web browser is the source in this case
- Registers to receive a notification when the user opens connect and selects a send target application.

Play Group
- Play group is displayed when source application supports playTo contract or Video Out is enabled

Send Group
- Send group is displayed when source application supports send contract

Figure 1: A source app participating in the sharing contract

When the user chooses the Connect Charm from the Windows Edgy bar, the Connect flyout opens and displays Send group in response:

1. Source applications provide the send-able item, potentially in multiple formats. The send-able item is also used to determine which target applications to display in the list. Items are excluded if they cannot support at least one of the formats for the send-able item.
2. Applications are displayed based on whether their package manifest indicates that they support send.
3. After a user chooses an application from the list, the target app flyout opens. Windows launches the correct target application and the target application displays UI for send. The target application reads the send-able item from the source application.

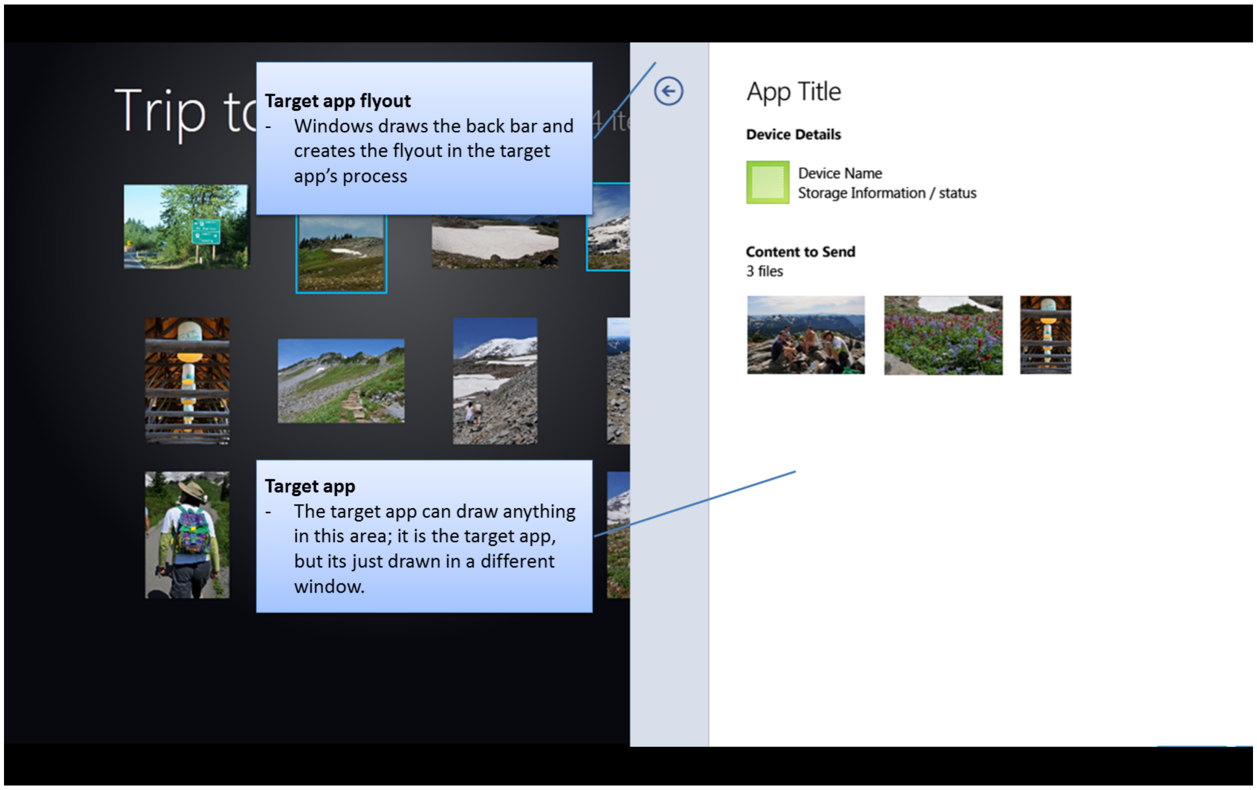

Trip to
Target app flyout
- Windows draws the back bar and creates the flyout in the target app's process
App Title
Device Details
Device Name
Storage Information / status
Content to Send
3 files
Target app
- The target app can draw anything in this area; it is the target app, but its just drawn in a different window.

Figure 2: A target app participating in the sharing contract

4.18.1 OVERVIEW OF THE SEND SOURCE APP CONTRACT

The Send source app contract is very similar to the Share source app contract. However, because Share is designed to work around people, it has additional user interface affordances that Send does not have. For example, the DataTransferManager object is common between Share and Send. However, the default behavior for Send is different (there is none) and the QuickLink mechanism is not used in Send.

When a source application implements the Send source app contract, it has to:

1. Create **DataTransferManager** object & handle **DataRequested** notification
2. If you're retrieving data synchronously, set the **Data** or invoke the **FailWithDisplayText** method on the **DataRequestedEventArgs** object before the timeout
3. If you're retrieving data asynchronously, then retrieve a **DataRequestDeferral** object and call Complete after the data is set or after there's an error getting the data

The **Data** should have metadata properties:

Property	Description
Title (required)	A string for title of send-able content

Description (optional)	A string for brief description of content

4.18.1.1 THE SEND SYSTEM FLOW – SOURCE APP

1. The user opens the source application. The source application registers to support send by creating a **DataTransferManager** object.
2. When the user invokes Connect from the Windows Edgy bar, the system asks source application for the send-able item if the application has registered for send.
3. The system uses the formats of the send-able item to filter the target applications.
   a. Note: If the source app is going to use delayed rendering of FileItems, the system uses the file extensions provided in the **FileTypes** property to filter target applications
4. The user chooses a target application. The system adds the applications to the OS Task completion list so they cannot be killed while participating in the send flow. The **TargetApplicationChosen** notification is sent so the source application can record the app for business intelligence purposes.
5. *<See steps the next section for information about what the flow after the user chooses a target application.>*

4.18.1.2 WINRT API DETAILS

The source contract for the Send flows and Sharing utilize the same WinRT objects. For charms invocation, the source application doesn't know whether it was invoked for Send or Sharing. For programmatic invocation, the source application has to choose which UI to display.

Windows.UI.Immersive.DataTransfer.DataTransferManager Static Object – used for programmatic invocation only [optional]

Methods	**Description**
ShowShareUI	Programmatic invocation of the Share flow. Most source applications should not need to use this because they can rely on the charms invocation.
ShowSendUI	Programmatic invocation of the Send flow (in the Connect charm). Most source applications should not need to use this because they can rely on the charms invocation.
GetForCurrentView	Returns DataTransferManager object associated with the current window.

Windows.UI.Immersive.DataTransfer.DataTransferManager Object

Events	**Description**
DataRequested	This notification occurs when the user presses Connect charm or Share

	(the source application intentionally does not know which charm the user pressed). The app needs to respond within the timeout by calling a method or setting the property on the *DataRequest* object retrieved through the *DataRequestedEventArgs* object.
TargetApplicationChosen	This notification occurs when the user chooses a target application in the Connect charm or Share. The app can use this to record information for business intelligence purposes about which applications are used to complete actions.

Windows.UI.Immersive.DataTransfer .DataRequestedEventArgs Object

Property	**Description**
Request	Allows developer to get the *DataRequest* object to either set the data or fail with display text.

Windows.UI.Immersive.DataTransfer.DataRequest Object

Methods	**Description**
FailWithDisplayText	Needs to be called before timeout in response to the event notification, *DataRequested*, to opt-out of Send or Sharing given the current state of the source application and provide an error string instead.
GetDeferral	Returns *DataRequestDeferral* object which is necessary in order to set data for share that is retrieved via an asynchronous operation.
Property	**Description**
Data	Needs to be set before timeout in response to the event notification, *DataRequested*, to provide a Data Package for Send or Share.

Windows.UI.Immersive.DataTransfer.DataRequestDeferral Object

Methods	**Description**
Complete	Is used in scenarios when data for send or share is retrieved via an asynchronous operation. After the data is set or an error occurs, call Complete.

Windows.UI.Immersive.DataTransfer.TargetApplicationChosenEventArgs Object – used for gathering business intelligence about the target application only [optional]

Property	**Description**
ApplicationName	After the event notification, *TargetApplicationChosen*, this property is set with the target application's name.

4.18.2 OVERVIEW OF THE SEND TARGET APP CONTRACT

To be a target application for Send, an application must:

1. Register support for the **SendTarget** extension in package manifest and include at least one supported file extension or data package format in application manifest (hint: file extensions must come first!).
2. Handle Send Activation & use the *Data* provided on the **SendTargetActivationContext** object. **SendTargetActivatedEventArgs**, which is passed through onActivate(), contains a property called *SendTargetActivationContext* which is of type **SendTargetActivationContext**.
3. For an extended (Long Running) send only, call **ReportExtendedSendStatus** once the user has committed the send task and the app is ready to be added to the send progress list in Connect Charm.
    a. Optionally report additional status to optimize resource usage of the system.
    b. Optionally call **SetExtendedSendDisplayText** to be shown on the app tile in the progress list.
    c. Optionally report a catastrophic failure with **ReportExtendedSendError**.
4. Call **ReportCompleted** when the send is complete if the send was successful

4.18.2.1 THE SEND SYSTEM FLOW – TARGET APP

1. The user installs the target application. The target application's package manifest declares that it supports send.
2. The system copies this information and stores it so it can be later queried to display applications in the send flyout.
3. *<See steps above in the system flow as it pertains to the source application>*
4. When the user chooses the target application, the system will activate that application via its application interface and using the details it provided during registration to launch it directly to its send view
    a. While the application is being activated, the system will display a splash screen for the application
5. The system will add the target application to the OS Task completion list so it cannot be killed while part of the send flow or when completing a send in the background
6. If the target application handles extended send—usually the case if they upload content types that can take more than a few seconds—they report status as they go:
    a. They report when the user presses the Submit/Send/Record button and thus if the user taps outside of the flyout, the system will hide the window vs. destroy it
7. When the target application is finished, it calls ReportCompleted. At this point, the system removes it from the OS Task completion list.

Note: There are optimizations the target can do to preserve battery life and these are covered in more detail in the Sharing Platform functional spec. They only apply to target applications that support extended send scenarios.

The following example shows how an application registers for send in its package manifest.

```
<Extension Category="windows.sendTarget" StartPage="device.html">
          <SendTarget>
            <SupportedFileTypes SupportsAnyFileType="false">
              <FileType>.png</FileType>
              <FileType>.customFileType</FileType>
            </SupportedFileTypes>
            <DataPackageFormat>Uri</DataPackageFormat>
            <DataPackageFormat>Text</DataPackageFormat>
          </SendTarget>
</Extension>
```

4.18.2.2 WINRT API DETAILS

Windows.UI.Immersive.SendTarget.SendTargetActivationContext Object

Property	**Description**
Data	A Modern Data Package that contains the data passed by the source application.
Methods	**Description**
ReportCompleted	The target application should call this after the send has completed successfully or unsuccessfully. The complete call closes the UI and effectively shuts down the application. A quick send is a task where it's reasonable for the user to view the UI until the send completes, such as send text/URI to device. For an extended send, this will be called after the final data transfer/send is complete, which is likely quite a while after the user presses 'send'. An **extended send** is a task where it's unreasonable to make the user wait and view the send UI until the send completes, because it will take more than 5 seconds, such as sending movies to a DVR.
ReportExtendedSendError	The target application should call this after the send has completed unsuccessfully and it has the option of returning a string that will be displayed in the UI. Generally, the application should report an error only if something unrecoverable has happened
ReportExtendedSendStatus	If the target application supports **extended send** scenarios, then it should use this method to report status to the send platform. The application needs to report values defined in the *ExtendedSendStatus* enum: • *Started* – This is *required* for extended send to behave

	correctly in the UI model. The application should report this after the user has committed the send task. This puts the app in the send progress list and enables a user to close the window without shutting down the application and thus stopping the send task. • *DataExtracted* – This is *optional* for extended send to optimize resource usage of the system. Applications should report this if they have finished extracting data from the Data Package. • *SubmittedToBackgroundManager* – This is *optional* for extended send to optimize resource usage of the system. Applications should report this if they have handed off the remainder of the send task to the Background Manager. An example is the WPD (Windows Portable Device) APIs.
SetExtendedSendDisplayText	The target application can call this while it is running to modify the string that is displayed in the background progress window. This is how an application can call the user's attention to take action in their application.

4.18.3 SAMPLE JAVASCRIPT CODE

The following code runs on the source app.

```
// Simple send scenario
var dtm =
Windows.UI.Immersive.DataTransfer.DataTransferManager.getForCurrentView();
dtm.addEventListener("datarequested", function (ev) {
    // Populate the data package properties
    ev.request.data.properties.title = "Title"; // required
    ev.request.data.properties.description = "Description"; // optional

    // Populate data package format(s), this is send a text item
    ev.request.data.setText("http://www.buildwindows.com");
});
```

The following code runs on the target app.

```
    // Initialize the activation handler
    Windows.UI.WebUI.WebUIApplication.addEventListener("activated",
activatedHandler, false);


    // global variable to store the SendTargetActivationContext object
    var sendCtx = null;
    // Handler executed on activation of the target
    function activatedHandler(eventArgs)
    {
        // in this sample we only do something if it was activated with
the Send contract
        if (eventArgs.contractId == "Windows.SendTarget")
            {
             // we receive the Send activation context as part of the
eventArgs
             sendCtx = eventArgs.sendTargetActivationContext;
            }
    }

        // Call the reportCompleted API on the global context we got from
the activation context
        sendCtx.reportCompleted(null);
```

4.18.4 REFERENCES

1. Connect Charm - M3 Functional Spec
2. SendTo Functional Spec
3. Building an application using send – UX guidance
4. Share team's documentation
    a. Share M3 User Experience Functional Spec, Share M3 Platform Functional Spec
    b. Share API Review Doc
    c. Building an application using share – UX guidance
    d. Code Samples

4.19 CONTROLS

Windows 8 features a new set of user interface controls that help you create apps with the look and feel of Windows in a way that is easy, fast, and hassle-free for both WWA and XAML development. You should use these controls when possible to achieve common Windows 8 UX Patterns rather than trying to replicate the pattern on your own through custom code. Here is a quick summary table of them:

Control Name	Category	Description	WWA API	XAML API
Flip View	Collection	Let's you flip through a collection of items, one at a time.	Win.UI.FlipView	<FlipView>
Grid View	Collection	Let's you view items in a 2D grid that scrolls	Win.UI.ListView *(with layout: { type: Win.UI.GridLayout})*	<GridView>

		side-to-side.		
Grouped Grid View	Collection	Let’s you view items in a 2D grid that scrolls side-to-side, where the items appear in groups.	Win.UI.ListView *(with layout: { type: Win.UI.GridLayout})*	<GridView IsGrouping = ' true'>
List View	Collection	Let’s you view items in a vertical list that scrolls up/down.	Win.UI.ListView *(with layout: { type: Win.UI.ListLayout})*	<ListView>
Semantic Zoom	Collection	Let’s you semantically zoom between two collection views.	Win.UI.SemanticZoom	<JumpViewer>
Web View	Content	Hosts web content.	<iframe>	<WebView>, <WebViewBrush>
Checkbox	General Controls	Represents a control that a user can select or clear.	<input type="check">	<CheckBox>
Date Picker	General Controls	Set a date	Win.UI.DatePicker	n/a
Hyperlink	General Controls	A button that appears as marked up text, typically used inline within text blocks.	<a>	<HyperlinkButton> <Hyperlink>
List Box, Drop-down	General Controls	Display a string with an dropdown icon next to it. Clicking it displays a popup list of items for a user to select among. The string shows the currently selected item.	<select>	<ComboBox>
List Box, Inline	General Controls	Displays an inline list of items for a user	<select>	<ListBox>

		to select among.		
Progress Bar	General Controls	Determinate and indeterminate progress bar control.	<progress> (width != height)	<ProgressBar>
Progress Ring	General Controls	Determinate and indeterminate progress ring control.	<progress> (width != height)	<ProgressRing>
Push Button	General Controls	A standard button that fires a click in response to a user tapping on it.	<button>	<Button>
Radio Button	General Controls	Allows a user to select a single option from a list of options. When radio buttons are grouped together they are mutually exclusive.	<input type="radio">	<RadioButton>
Rating	General Controls	View or edit star ratings	Win.UI.Rating	n/a
Slider	General Controls	Slider	<input type="range">	<Slider>
Time Picker	General Controls	Set a time	Win.UI.TimePicker	n/a
Toggle Switch	General Controls	Switch settings on and off	Win.UI.ToggleSwitch	<ToggleSwitch>
Audio Playback	Media	Plays audio (with playback controls). Supports DRM and non-DRM.	<audio controller=true>	<MediaPlayer>
Video Playback	Media	Plays video (with playback controls). Supports DRM and non-DRM.	<video controller=true>	<MediaPlayer>
App Bar	Overlay	Application toolbar for displaying	Win.UI.AppBar	<ApplicationBar>

		commands.		
Context Menu	Overlay	Custom context menu presenting commands specified by app developer.	Windows.UI.PopManager	n/a
Flyout	Overlay	Flyout	Win.UI.Flyout	n/a
Tooltip, Rich	Overlay	Tooltip that can support rich content (e.g. images and formatted text) to show more information about something.	Win.UI.Tooltip	<Tooltip>
Tooltip, Simple	Overlay	Plain text tooltip to show more information about something.	<div title='foo'>	<Tooltip>
Multi-line Text Box	Text	Mutli-line plain text box.	<textarea>	<RichEditBox>, <TextBox AcceptsReturn = 'true'/>
Password box	Text	Password box	<input type="password">	<PasswordBox>
Rich Text Box	Text	Multie-line rich/styled text box.	<div contentEditable=true>	<RichEditBox>
Single-line Text Box	Text	Single-line plain text box	<input type="text">	<TextBox>, <RichEditBox AcceptsReturn= 'false'>
Panning ScrollView	View	Displays a view of content a user can pan around in one or two dimensions, and various features like snap points, etc. that enhance that experience.	<div style=overflow:scroll>	<ScrollView>
Scroll Bar	View	Not directly exposed to	<div style=overflow:scroll>	<SrollBar>

		developers. It manifests itself in the context of scroll views. You can't just include it as a standalone control in an app.		
Zooming ScrollView	View	Displays a view of content a user can zoom in/out of, and various features like snap points, etc. that enhance that experience.	<div style=overflow:scroll; -ms-content-zommable:true>	<ScrollView>

For a full list of controls with additional details, see Windows UI Toolkit for Tailored Apps. For guidelines on how to best use the different controls, see Designing and Building an App with the Windows 8 Look and Feel.

The next sections describe the controls programming model and the specific controls in more detail.

4.19.1 USING CONTROLS

4.19.1.1 USING CONTROLS (WWA DEVELOPMENT)

To get started using controls you need to include the appropriate CSS and JS files from the Windows Library for JavaScript (a.k.a. WinJS) into your project. The default project templates in Visual Studio already include these, but for your reference these are at a minimum the files you need to include:

```
<link rel="stylesheet" href="/winjs/css/ui-dark.css" />
<script src="/winjs/js/base.js"></script>
<script src="/winjs/js/ui.js"></script>
<script src="/winjs/js/binding.js"></script>
<script src="/winjs/js/controls.js"></script>
<script src="/winjs/js/animations.js"></script>
<script src="/winjs/js/uicollections.js"></script>
```

Note that it's important to include the *ui-dark.css* stylesheet not only for controls, but also because it provides an overall base default style for your application including page background, typography, etc.

The individual controls you may want to use fall into two categories:

1. **HTML Controls.** These are built directly into HTML and typically map to W3C standards such as <button>, <input type="range">, <div overflow:scroll>, and <video>. While the tags, methods, attributes, etc. to program against them are standard, their implementations and their look and feel have been updated to match that of Windows 8.

   Here's an example of how you would instantiate a standard progress control declaratively in HTML markup:

```
<progress id='myProgress' value='75' max='100'>
```

2. **JavaScript Controls.** These controls were created specifically for WWAs and go beyond web standards. Windows 8 defines a common JS control model for instantiating and using these controls. These controls are available for WWAs through the Windows Library for JavaScript under the **Win.UI** namespace (note: PDC-3 and beyond it will be the **WinJS.UI** namespace).

   To instantiate controls *declaratively* you simply include inline markup right in your HTML such as the following example for the ratings control, specifying the name of the control type and any optional parameters:

```
<div class="myRating" data-ms-control="Win.UI.Rating"
    data-ms-options="{maxRating: 5, averageRating: 3.5}">
</div>
```

   If using the above declarative method you also need to include a call (typically in response to the DOMContentLoaded event) to the following function to actually do the instantiation:

```
WinJS.UI.processAll();
```

   If you prefer to do the instantiation manually you can do it this way and you don't even need to use the declarative markup:

```
var progress = new Win.UI.Rating(
    document.getElementById('myRating'), {
    maxRating: 5,
    averageRating: 3.5}
);
```

In addition to the Standard and Javascript based controls, there are some system features that people sometimes think of as controls, most notably the context menu, which are exposed as WinRT API projections. See WinRT documentation for information on instantiating those.

4.19.1.2 USING CONTROLS (XAML DEVELOPMENT)

For XAML-based development, all controls are standard XAML elements. See XAML documentation for more details.

4.19.1.3 REFERENCES

1. Windows UI Toolkit for Tailored Apps
2. A Developer's Guide To Consuming PAC Web Technologies
3. WinRT UI development cookbook
4. JS Control Model Spec
5. Corsica Object Design Walkthrough
6. W3C HTML 5 Spec

4.19.2 LISTVIEW

A very common and important scenario in MoSh applications is presenting a collection of items to users. The **ListView** is a WinUI JavaScript control that enables easy creation and management of a data collection in a WWA and provides an interaction model consistent with that of Windows.

It is possible to virtualize, data bind, and template items in the **ListView** by passing a data source and an item renderer to the control. The **ListView** control also provides a set of options for altering its look and feel, and it supports the styling of items via CSS classes.

To use the **ListView** in your Windows Web Application, include uicollections.js and uicollections.css as part of your application.

4.19.2.1 API DETAILS

ListView Object

Properties	**Description**
layout	Specifies the layout of the **ListView**. While custom layouts are possible, the most common values are the pre-defined layout objects `{type: Win.UI.GridLayout}` and `{type: Win.UI.ListLayout}`
selectionMode	Specifies the selection mode of the **ListView**. Possible values: **none**, **single**, **multi**, **extended**.
tap	Specifies what should happen when the user taps on an item. Possible values: **selectandinvoke**, **invokeonly**, **none**
crossslide	Specifies what should happen when a user crossslides on an item. Possible value: **select**, **none**
loadingBehavior	Controls the loading behavior of the **ListView**. Possible values: • **randomaccess** The scrolling region of ListView is calculated based on the entire set of data. Items will be swapped in and

	out of the DOM such that at any given time the items that exist in the DOM are the current viewport items and one page before and after the viewport. • **incremental** The **ListView** loads items in chunks. When a user reaches a threshold scrolling amount in the scroll region of the **ListView**, the next chunk of items is loaded into the DOM.
automaticallyLoadPages	When the loadingBehavior is incremental, this determines if new pages should be loaded automatically. Possible values are **true** and **false**.
loadNextPages	Manually initiates an incremental load. The number of pages to be loaded is determined by pagesToLoad (see below).
pagesToLoad	Determines how many pages of items should be loaded when an incremental load happens.
pageLoadThreshold	When automaticallyLoadPages is true, an incremental load will be automatically initiated when the user is **pageLoadThreshold** away from the end of the list.
reorder	Specifies the reorder mode of the **ListView**. Possible values: **true**, **false**.
editable	Can be set to **true** or **false**. If false, the ListView will disable CUT and DELETE operations by the user.
scrollPosition	The position of the scrollbar in the ListView.
itemRenderer	Specifies the function that the Items Manager uses to create item templates. This function can be a Corsica data template, or a custom rendering function of the signature: "function (item)." Setting this property is required if you want to style the **ListView** items.
dataSource	Specifies the data source used by the ListView to generate items. Data sources can be created using the following functions: **createObjectDataSource**, **IteratorDataSource**, **createVectorDataSource**.
groupDataSource	Specifies the data source for Groups. The second parameter of the GroupDataSource object is a function that describes the groupBy function. For example:

```
listView.groupDataSource = new
Win.UI.GroupDataSource(listView.dataSource, function
(item) {
    var firstLetter = item.title.toUpperCase().charAt(0);
    return {
        key: firstLetter,
        data: {
            title: firstLetter
        }
    };
```

	`});` Setting this property is required if you want to enable groups.
groupHeaderPosition	Specifies the position of group headers. This is a property of the **layout** property (which in turn is a property of the ListView—see above). Possible values are **"top"** and **"left"**.

groupRenderer	Specifies the function that determines the look of the group header. The function has the following signature: "function (item)," where item is group for which the function should generate an element. For example:
	
	`listView.groupRenderer = function (item) {`
	`    var div = document.createElement("div");`
	`    div.innerText = item.data.title;`
	`    return div;`
	`};`
	
	Setting this property is required when using groups.

Events	**Description**
itemInvoked	Event raised when the user clicks on item in browse mode.
selectionChanging	Event raised before selection changes. To cancel the selection change, set the **Allowed** flag to **False**. To cancel selection, applications should call **preventDefault** instead of setting the **Allowed** flag.
selectionChanged	Event fires when selection has changed.
readyStateChanged	Event raised when the view state changes during initialization and after scrolling operations. Two values are reported: "loading" which is equivalent to ListViewState.initalized. "complete" which is equivalent to ListViewState.ready.
dragitemsstart	Event raised by the drag-and-drop source when a new drag-and-drop operation is initiated.
dragitemsend	Event raised by the drag-and-drop source when the user drags to a desired drop target.
dragitems	Event raised by the drag-and-drop source when any items are dragged.
dragitemsenter	Event raised by the drag-and-drop target when any items are dragged into a desired drop target.
dropitems	Event raised by the drag-and-drop target when any items are dropped into a desired drop target.
Methods	**Description**
addEventListener	Implements the DOM level 2 EventTarget interface method.
ensureVisible	Makes the item at the specified index of the data set visible within the viewport.
firstVisible	Retrieves the index of the first visible **ListView** item.
getElementAtIndex	Retrieves the index of the **ListView** item.

getIndexFromElement	Retrieves the HTML node at the specified index.
lastVisible	Retrieves the index of the last visible item in the viewport.
removeEventListener	Implements the DOM level 2 EventTarget interface method.
scrollTo	Scrolls the **ListView** viewport to the specified index.
selection	Gets or sets the selected items for the **ListView**. Specifying no parameters returns the currently selected items. Specifying an array of integers sets the **ListView** objects selected items.

Datasource Object (listview.datasource)

Methods	**Description**
createListBinding	Creates a ListBinding that can be used to retrieve item information from the dataSource. A binding must be created in order to retrieve actual item data from the dataSource. See the ListBinding Object methods below.
refresh	Forces a refresh.
beginEdits	Notifies the Items Manager that a sequence of edits is about to begin. It should be called immediately before edits are made to the list.
getCount	Retrieves the number of items in the data source's data set.
remove	Removes the item at a specified **key**.
endEdits	Notifies the Items Manager that a sequence of edits is about to end. It should be called immediately after edits are made to the list.
insertBefore	Signature is (key, data, nextKey). Inserts the specified **data** into the dataSource, with the specified **key**, before the item at **nextKey**.
insertAtStart	Signature is (key, data). Inserts the specified **data** into the dataSource, with the specified **key**, before the item at the start of the list.
insertAfter	Signature is (key, data, previousKey). Similar to insertBefore, except inserts after **previousKey**.
insertAtEnd	Signature is (key, data). Similar to insertAtStart, except inserts at the end of the list.
change	Signature is (key, newData). Provides new data for the item with **key**.
moveBefore	Signature is (key, nextKey). Moves the item with **key** to before **nextKey**.
moveToStart	Signature is (key). Moves the item with **key** to the start of the list.
moveAfter	Signature is (key, previousKey). Moves the item with **key** to after **previousKey**.
moveToEnd	Signature is (key). Moves the item with **key** to the end of the list.

ListBinding Object (retrieved via listview.datasource.createListBinding)

Methods	**Description**
first	Retrieves the first item in the list
last	Retrieves the last item in the list
fromKey	Signature is (key, prefetchBefore, prefetchAfter). Retrieves the item with the specified **key**. prefetchBefore and prefetchAfter are optional params.
fromIndex	Signature is (index, prefetchBefore, prefetchAfter). Retrieves the item at the

specified **index**. prefetchBefore and prefetchAfter are optional params. *(note: these are the main functions most people will use, but there are others; see References section, #3)*

4.19.2.2 INITIALIZATING THE LISTVIEW

To initialize a **ListView**:

1. In your HTML, create a **div** element that will be the parent of the **ListView** control. Make sure that you specify a height and width for this **div**.
2. In your code behind, create the **dataSource** that contains the data to display.
3. The example below uses Corsica templating to define how each item in the ListView will look. You can also pass your own templating function to the **itemRenderer** if desired.

The following example will create a **ListView** with a Corsica data template for each item in the view.

```
// Required JavaScript Files
<link rel="stylesheet" href="/winjs/css/ui-dark.css" />
<script src="/winjs/js/base.js"></script>
<script src="/winjs/js/ui.js"></script>
<script src="/winjs/js/binding.js"></script>
<script src="/winjs/js/controls.js"></script>
<script src="/winjs/js/animations.js"></script>
<script src="/winjs/js/uicollections.js"></script>

// HTML Markup
<div id="itemTemplate" data-win-control="Win.Binding.Template">
  <div class="imageWithTextTemplateClass">
    <div class="title" data-win-bind="innerText: title"></div>
    <div data-win-bind="innerText: text"></div>
  </div>
</div>


<div id="listView" data-win-control="Win.UI.ListView"
  data-win-options="{
      dataSource: myData,
      itemRenderer: itemTemplate,
      selectionMode: 'multi',
      layout: {type: Win.UI.GridLayout}
  }" >
</div>
```

4.19.2.3 SAMPLE JAVASCRIPT CODE

```
// An array of JSON data, which can be passed to the dataSource property
// to provide data to the ListView
```

```
var myData = [
  {title: "Vanilla", text: "Vanilla Ice Cream"},
  {title: "Strawberry", text: "Strawberry Sorbet"},
  {title: "Banana", text: "Banana Frozen Yogurt"}
];

// Process the Control
document.addEventListener("DOMContentLoaded", function () {
          Win.UI.processAll();
     }, false);
```

More examples and code can be seen in the QuickStart for ListView guide. (currently in draft state).

4.19.2.4 REFERENCES

1. ***Best Resource***: Send questions or comments to the **listviewdisc** alias!
2. ListView Sample Tests. A good place to peek at ListView test code; focus on SimpleTA.html.
3. Useful API Doc on DataSource APIs

4.19.3 FLIP VIEW

The **FlipView** is a collection control that displays a single item from a list at a time and provides simple forward and backward navigation. Just as for **ListView**, it is possible to virtualize, data bind and template items in the **FlipView** by passing a data source and an item renderer to the control.
The **FlipView** also supports styling of items via CSS classes.

4.19.3.1 API DETAILS

FlipView Object

Properties	**Description**
orientation	Specifies the orientation of the FlipView control: **horizontal** or **vertical**.
dataSource	Specifies the items to be displayed in the FlipView.
itemRenderer	Specifies the data templating function for the item manager.
placeholderRenderer	Specifies the templating function that generates placeholder items while the actual item loads.
currentPage	Gets and sets the current page that is being displayed in the FlipView.
Events	**Description**
PageVisibilityChanged	Raised when a page becomes visible or invisible. It is fired on the content element itself and bubbles up. The source property is the HTML element on which the FlipView is attached, and visible is a boolean that indicates whether the page is visible.
DatasourceCountChanged	Raised when items are changed, inserted, or added to the data source.
Methods	**Description**
next	Flips the FlipView to the next item.
previous	Flips the FlipView to the previous item.

count	Retrieves the number of items in the data source. This is done via a promise.
SetCustomAnimations	Takes an object that can have three properties: next, previous, jump. These properties are functions that have the signature: function(ingoingItem, incomingItem). This function is responsible for animating the two items in/out of the view. This function must return a WinJS.Promise, that should be completed when the function is done animating.

4.19.3.2 INITIALIZING A FLIPVIEW

To initialize a **FlipView**:

1. In your HTML, create a **div** element that will be the parent of the **FlipView** control. Make sure that you specify a height and width for this **div**.
2. In your code behind, create the **dataSource** that contains the data to display. Create an **itemRenderer** if you do not want to use default item template.

The following example initializes a **FlipView**.

```
var options = {dataSource: someDataSource, itemRenderer: someRenderer};
var listview =
Win.UI.Controls.FlipView(document.getElementById("someDiv"), options);
```

4.19.3.3 SAMPLE JAVASCRIPT CODE

```
// Note: Corsica Libraries must be included for this to function properly
<script type="text/javascript" language="javascript">
    function onLoad() {
    var FlipViewDiv = document.getElementById("FlipView");
    var FlipView = Win.UI.FlipView(FlipViewDiv, { dataSource:
          createSimpleDataSource(), itemRenderer: FlipViewItemRenderer });
    }

    function createSimpleDataSource() {
     var testData = [
{ name: "http://www.cbhbblog.com/wp-content/uploads/2010/09/MSU-Flag.jpg" },
{ name: "http://images.dawgsports.com/images/admin/MSU_mascot_helmet.jpg" },
{ name: "http://www.math.msu.edu/images/msu1.jpg" }];

     return new Win.UI.ArrayDataSource(testData);
    }

    function FlipViewItemRenderer(e) {
     var result = document.createElement("div");
     var image = document.createElement("img");
     image.src = e.data.name;
     result.appendChild(image);
     return result;
    }
    document.addEventListener("DOMContentLoaded", onLoad, false);
```

```
</script>

<div id="FlipView" class="FlipView"></div>
```

4.19.3.4 REFERENCES

1. FlipView API Documentation Wiki
2. FlipView API Spec
3. Items Manager API Documentation Wiki
4. Useful API Doc on DataSource APIs

4.19.4 APPLICATION COMMAND UI

Application Command UI enables applications to have fullscreen immersive experiences that prioritize content while allowing the applications to still have a reliable method for invoking hidden commands. Application Command UI can be invoked and dismissed using touch, mouse, and keyboard.

4.19.4.1 MECHANICS OF THE TOUCH GESTURE

If the user performs a single finger swipe that starts on the top or bottom edge of the screen and is perpendicular to the edge, then Application Command UI events are fired. Swipes that are parallel to the edge of the screen, taps, or multi-finger input are normal input that goes through to the app.

4.19.4.2 EVENTS FOR TOUCH

When an Application Command UI gesture occurs, the application receives a series of events. Applications should listen for these events and respond to them by showing secondary navigation, commands, and information. If the secondary commands are already shown, then a series of Application Command UI events should be interpreted as a dismissal of those app commands.

As soon as the user starts swiping for the gesture, an *invoking* event occurs, which is when applications should start animating in/out their command UI. Once the user completes a swipe, then an *invoked* event occurs, which is when the UI should conclude its entrance/exit. If the user swipes toward the middle of the screen, but then in the same gesture “cancels” out the gesture by swiping back to the edge of the screen, then a *cancel* event occurs; this should cause the Command UI to dismiss if it was in the process of animating in. Gestures starting over/under the Fill app go to the Fill app. Gestures starting over/under the Snapped app go to the Snapped app. The app is given activation before the first Application Command UI Invocation event is fired. No location information of the touch gesture is given. The same event is fired for gesture from the top and bottom edges. These events include a “source” attribute indicating whether they were triggered by an edge gesture or a keyboard shortcut.

Whenever an Application Command UI gesture occurs, then the Application Command UI events are always fired. It is up to the app to handle the events. There is no way for apps to opt out of app gesture detection.

4.19.4.3 EVENTS FOR MOUSE

Applications should show/hide their command UI when a right click event over the app is detected.

4.19.4.4 EVENTS FOR KEYBOARD

When the user presses Win+Z, the Application Command UI *invoked* event is sent to the app that has focus with the source property of keyboard shortcut.

4.19.4.5 SAMPLE JAVASCRIPT CODE

```
// Register for Application Command UI events
var appCmdUI =
Windows.UI.Immersive.ApplicationCommandUI.getForCurrentView();

appCmdUI.addEventListener("invoking", onInvoking);
appCmdUI.addEventListener("invoked", onInvoked);
appCmdUI.addEventListener("canceled", onCanceled);

// Register for Context Menu events
document.oncontextmenu = onContextMenu;

// JavaScript definition for right mouse button
var MOUSE_RIGHT_BUTTON = 2;

// Show Application Command UI on mouse right click or context menu key
function onContextMenu(e) {
    // determine whether it was mouse right click
    if (e.button == MOUSE_RIGHT_BUTTON) {
           alert("Right Click Invocation");
    }
}

// Show Application Command UI in an entry state
function onInvoking() {
    alert("Invoking with touch");
}

// Show Application Command UI in an invoked state
function onInvoked(e) {
    // determine whether it was touch or hotkey invocation
    if (e.source ==
        Windows.UI.Immersive.ApplicationCommandSource.edgeGesture)
  {
           alert("Invoked with touch");
    }
    else if (e.source ==
            Windows.UI.Immersive.ApplicationCommandSource.keyboardShortcut)
    {
            alert("Invoked with keyboard shortcut");
    }
}

// Dismiss the attempt to invoke Application Command UI. If the
// invoking handler changed the UI state (e.g. showing an overstretched
command bar), dismiss it here.
function onCanceled() {
    alert("Canceled with touch");
}
```

4.19.4.6 REFERENCES

1. Edge UI Invocation Functional Spec

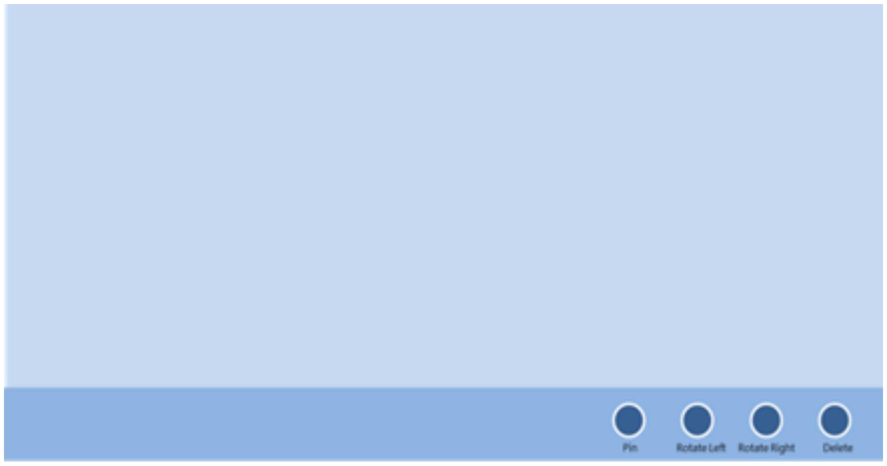
4.19.5 APPLICATION BAR

The application bar is an in-framework control that provides a common way for applications to present navigation, commands, and tools in a manner consistent with the content over chrome philosophy (reduce distractions to provide an immersive experience). The application bar uses Windows styling, but is customizable.

The application bar is an application's primary UI surface for commands. If users must have access to commands that are not provided by direct manipulation and are not critical on-canvas commands, then add those commands to an application bar.

There are two styles of application bar, the transient bar and the persistent bar. Transient bars start out hidden. The user can show or hide the bar by swiping on the top or bottom edge of the screen. Transient application bars can also be set to dismiss when the user interacts with the application or to dismiss after a period of time. Persistent bars remain on screen though the user can control, via an edge swipe, whether the button labels are shown or hidden.

If users must use the commands on the application bar to complete their primary task, then the bar should be persistent. If the commands are secondary to the user's task, then they should be on a transient bar.



Application bar with 4 commands

4.19.5.1 SAMPLE JAVASCRIPT CODE

The following example shows how to initialize an **application bar**:

```
// Note: Corsica Libraries must be included for this to function properly

<script type="text/javascript">
     // Appbar initialization
     document.addEventListener("DOMContentLoaded", function () {
         Win.UI.processAll();
     }, false);
</script>
```

This example uses HTML to create a lightweight, transient **application bar**:

```
<!-- Bottom Command Bar -->
    <div data-win-control="Win.UI.AppBar" data-win-
options="{position:'bottom',transient:true,autoHide:0,lightDismiss:true}">
        <div class="win-appbar-commands">
            <!-- Add Content Here -->
            <button onclick="alert('Command 1')">
                <img src="1.png" alt="Command 1" /><span class="win-
appbar-expands">Command 1</span>
            </button>
            <button onclick="alert('Command 2')">
                <img src="2.png" alt="Command 2" /><span class="win-
appbar-expands">Command 2</span>
            </button>
        </div>
    </div>
```

This example shows how to programmatically close the application bar programmatically.

```
// This is how we would hide our bar programmatically (event will do it
// automatically)
document.getElementById("myTopBar").msControlObject.hide();
```

4.19.5.2 REFERENCES

1. App Bar M3 Functional Spec
2. App Bar M3 Dev Design
3. App Bar M3 DDI Spec

4.19.6 SPLASH SCREEN

The splash screen is a required visual component used by the system to provide instant feedback to users when your application is launched. For all applications, the splash screen is shown briefly while your application is initialized, affording a clear indication that a transition has occurred and that the application will soon take control of the experience. Once your application returns from activation and is ready for interaction, the splash screen will automatically dismiss.

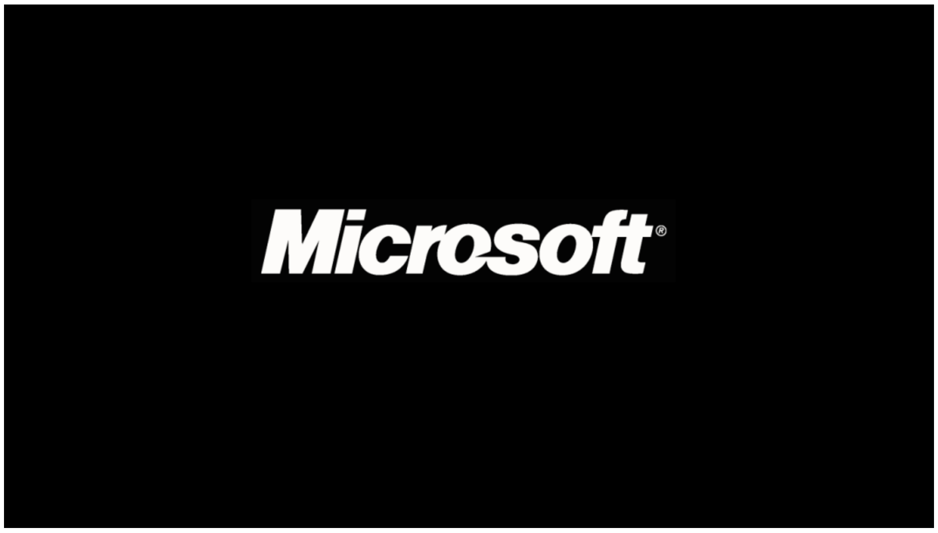
All applications must declare a splash screen. To customize the splash screen, developers modify the splash screen element in the application's package manifest. This element consists of two attributes:

- **Image (Required)**
  The image must be a 624x304 JPG or PNG and can be localized for different languages and/or contrast modes. To accommodate scaling, two versions of each image asset must be provided.
- **Background Color (Optional)**
  The background of the splash screen is a solid color specified using hexadecimal color codes or one of 16 standard color strings defined by HTML5. If omitted, the splash screen will inherit the value of the background color attribute of the VisualElements node.

The following example demonstrates the declaration of the splash screen in the manifest:

```
<SplashScreen BackgroundColor="black" Image="splash.png" />
```

Using the above example, the splash screen would consist of a black background and the image splash.png.



4.19.6.1 SPLASH SCREEN FLOW

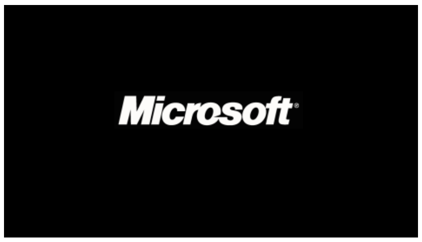
1. The developer adds the splash screen registration markup to the application's manifest. This registration includes details about what content will be displayed in the splash screen.
2. The user installs the application.
3. Upon install, Windows stores the splash screen content details from the app's manifest.
4. The user launches the application.
5. The splash screen is immediately triggered and shown for a minimum of one second (for the duration of the launch animation).
6. At that point, if the application is ready the splash screen will be dismissed promptly. Otherwise, the splash screen will remain on screen until the app returns from activation or a timeout occurs (15 seconds).
7. The splash screen is dismissed by the system using a brief cross-fade animation.

4.19.6.2 SPLASH SCREEN API

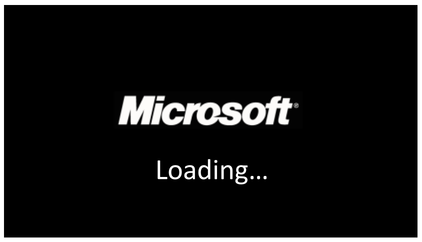
In addition to specifying the static content to be displayed within the splash screen, you can also utilize the splash screen API to further customize your application's loading experience. Because many applications will need to complete additional loading tasks upon startup, the splash screen API offers a way to extend the splash screen experience into your application, creating one seamless loading flow. In addition, the splash screen API also provides a way for applications to register to be notified when the splash screen has been dismissed. Applications will use this notification to determine when it is appropriate to begin content entrance animations or other operations once the application's UI is completely in view.

4.19.6.2.1 EXTENDING THE SPLASH SCREEN

To extend the splash screen experience into the application's first view, applications can retrieve the splash screen image coordinates from the **SplashScreenInfo** object, located off of the activation eventargs. Using these coordinates, applications can position the image in the same location of their first view. This way, the transition from the splash screen to your application will be abstracted from the user. Because your application will be in complete control of the experience at that point, it is possible to provide additional loading information or progress before you are ready to navigate to your application's landing page.



System splash screen



Application's first view with additional loading text

4.19.6.2.2 REGISTERING FOR DISMISSAL NOTIFICATIONS

To listen for the splash screen dismissal notification, applications can register for the **splashscreendismissed** event, located off of the activation eventargs. This event fires when the splash screen has been fully dismissed, indicating that it is safe to start any operations that require the application's UI to be in view. For example, it is recommended that applications handle this event to start any content entrance animations that may be used.

4.19.6.3 SAMPLE JAVASCRIPT

Extending the splash screen:

```
Windows.UI.WebUI.WebUIApplication.addEventListener("activated",
onActivation, false);

function onActivation(e){
    var splash = e.splashScreenInfo;

    // Retrieve the window coordinates of the splash screen image
    coordinates = splash.imageLocation;

    // Position the extended splash screen image in the same location
    var image = document.getElementById("extendedSplashImage");
    image.style.position = "absolute";
    image.style.left = "" + coordinates.x + "px";
    image.style.top = "" + coordinates.y + "px";
}
```

In the above example, “extendedSplashImage” would refer to an <img> element in the application’s first view.

Registering for dismissal notifications:

```
Windows.UI.WebUI.WebUIApplication.addEventListener("activated",
onActivation, false);

function onActivation(e){
    var splash = e.splashScreenInfo;

    // Register an event handler to be executed when the splash
    // screen has been dismissed
    splash.addEventListener("splashscreendismissed", onSplashDismissed,
false);
}

function onSplashDismissed() {
    // Start content entrance animations...
}
```

4.19.6.4 REFERENCES

1. Splash Screen M3 Functional Spec
2. Splash Screen M3 API Summary
3. Sample Splash Screen Image

4.19.7 MESSAGE DIALOG

A message dialog presents the user with an urgent message or a question that must be answered. It can be dismissed programmatically, or when the user chooses a button. Message dialogs are UI modal – the user cannot interact with the application that launched the message dialog until the dialog is dismissed. Message dialogs are asynchronous – the code that calls the message dialog is not blocked and will continue to execute. Callbacks are used to know when a button is pressed or the dialog is dismissed. Dialogs appear vertically centered and fill the entire width of the screen.

Message dialogs are designed to be intrusive so that the user must acknowledge its content before continuing on in their immersive experience. They should be used sparingly since they are very disruptive for the user.

A message dialog can contain 3 pieces of information:

- Title (optional)
- Content string (required)
- Up to 3 buttons (optional) – if no button is specified a default Close button is provided

The title is used to give context to the message that is being presented and they should only be used if necessary. Most of the time, a message dialog is just the content string is appropriate. If used, the title should be kept very short, and not contain any instruction or question.

The content string contains the main instruction or question of the dialog.

The buttons should be used to answer the specific question and should not be generic **OK**/**Cancel** buttons. In the preceding example, the two buttons are labeled as **Allow** and **Don't Allow** because they provide concrete answers to the question the dialog asks.

4.19.7.1 SYSTEM FLOW

1. The user launches a modern application.
2. The user takes some action that requires a message dialog.
3. The application creates the MessageDialog object and sets the content string and, if necessary, title.
4. The application adds any buttons, and specifies the appropriate callbacks that are necessary to the dialog by calling **MessageDialog.commands.append(UICommand)**;
5. The application shows the MessageDialog using MessageDialog.ShowAsync().operation.start() or optionally shows it and specifies a callback for when the dialog is closed by using MessageDialog.ShowAsync().then(callback).
6. The system shows the message dialog that contains the strings and buttons that the application specified.
7. The user selects a button from the message dialog.
8. The appropriate callback is called.

4.19.7.2 WINRT API DETAILS

MessageFlyout Object	
Property	**Description**
title (optional)	A short string that offers context for the content string
content (required)	A string that represents the primary piece of information or question being communicated by the message flyout.
commands (optional)	A standard vector, for the developer to add or remove commands (UICommand) to and from. If none is specified, a Close button is provided.
Methods	**Description**
showAsync()	Creates the Async Object that is used to show the message dialog asynchronously.

UICommand Object	
Property	**Description**
Label	The name of the specified command that will be displayed in the UI.
Action	The callback that is called when the user presses a button.

Async Object	
Method	**Description**
then(callback)	Shows the message dialog, and calls the optional callback when the dialog is closed for any reason.
Property	**Description**
operation	The Operation Object that represents the asynchronous operation of showing a message dialog. Contains the **start()** method which shows the message dialog.

4.19.7.3 SAMPLE JAVASCRIPT CODE

The following example creates a message dialog that contains a message. It contains the default Close button and specifies a callback for when the dialog is closed.

```
var msg = new Windows.UI.Immersive.MessageDialog("Your trial has expired.
You must upgrade to continue using this app. ");

msg.showAsync().then(function () {
    //Navigate to the upgrade page when the user presses Close
});
```

The following example creates a message dialog that contains a title, content, and custom buttons with callbacks.

```
var msg = new Windows.UI.Immersive.MessageDialog("Trial Expired", "To
continue using the app you can upgrade to either the basic or professional
versions. The professional version is more full featured than the basic
version. ");

msg.commands.append(new Windows.UI.Immersive.UICommand("Buy Basic",
function () {
    //Navigate to upgrade to basic page
}));

msg.commands.append(new Windows.UI.Immersive.UICommand("Buy Professional",
function () {
    //Navigate to upgrade to professional page
}));

msg.commands.append(new Windows.UI.Immersive.UICommand("Extend Trial",
function () {
    //Extend the user’s trial
}));

msg.showAsync().operation.start();
```

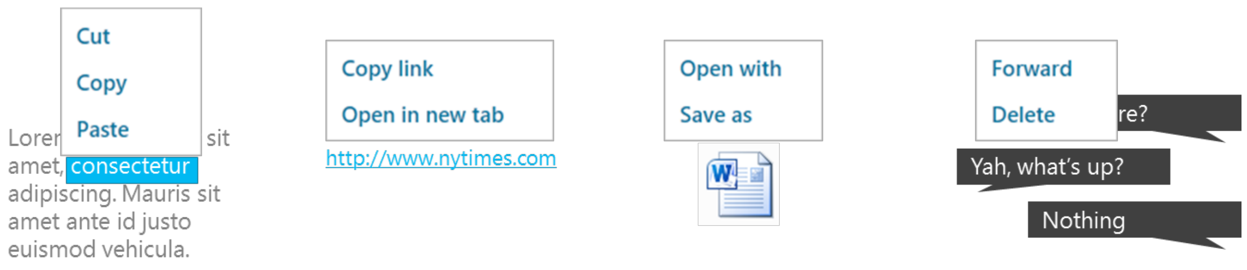
4.19.7.4 REFERENCES

1. Message Dialog API Summary
2. Message Dialog Functional Spec

4.19.8 CONTEXT MENU

The context menu is a lightweight control that is for performing lightweight actions on text; specifically, cut, copy and paste. It is great at showing a few commands and can be used for acting on an object without a selection, usually one thing. The context menu only has five slots so it cannot be used as the primary commanding surface for an application.

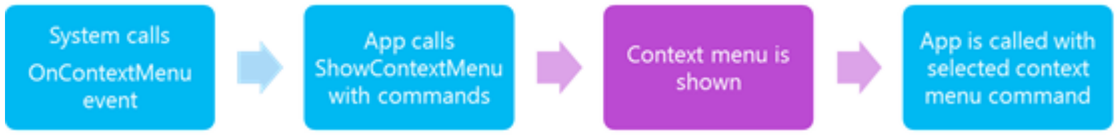
In general, apps should be rethought for touch in Windows 8. Most of the cases where commands in toolbars and context menus were used in the desktop, should be replaced with direct and immersive interaction. How could your command be re-envisioned and optimized for the form-factor? What parts of your application could be best at touch, how can these be at the center. Example: Allow a user to directly transform a picture with their fingers instead of introducing a “rotate” context menu command for “Rotate.”



How the Context menu works:

1. The user launches a modern application
2. The user selects an item or presses and taps on an item.
3. The system fires the **OnContextMenu** event.

4. The application receives the **OnContextMenu** event and creates the **IPopupMenu** object.
5. The application adds commands that apply to the selection by calling **Command.Append**. Applications can specify name of the command and the callback to be called when the menu item is clicked.
6. The applicant then specifies a rectangle that corresponds to the selection that the context menu should be shown near. Optionally, the application can specify where the context menu should appear relative to the select selection (the preferred side). The system may ignore the selection rectangle and preferred side to keep the context menu on screen.
7. The system shows the context menu.
8. The user selects a command from the context menu or dismisses the context menu by tapping outside of the context menu.
9. If a command was selected, the application is called back into with the selected command. Otherwise, the **onDismissed** event is raised.
10. When it handles the command callback, the application can display a flyout.


System calls OnContextMenu event
App calls ShowContextMenu with commands
Context menu is shown
App is called with selected context menu command

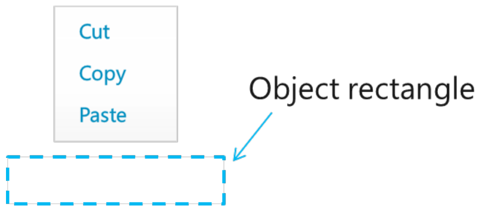
4.19.8.1 CONTEXT MENU USAGE

4.19.8.1.1 APPROPRIATE USE OF CONTEXT MENUS

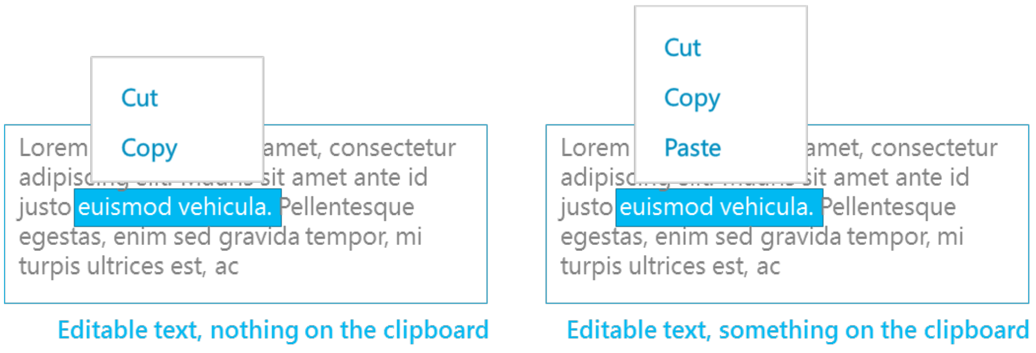
- **For Cut, copy and paste**
  If you have an object that supports being cut, copied or pasted to the clipboard use the context menu to show these commands. The context menu is the consistent way to perform clipboard actions in the Mosh.

- **If there is no way to indicate or create selection**
  If you have an object that needs to support commanding but it isn't possible to indicate or create selection you should use the context menu to show commands.

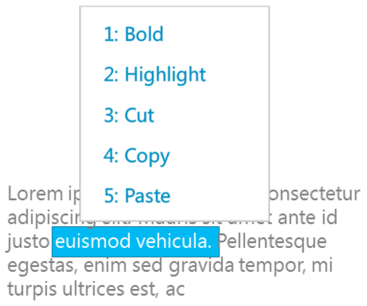
- **Positioning the menu**
  Position the context menu relative to an object using relative positioning logic.



- **Showing and hiding commands**
  Context menus do not have a disabled state; show and hide commands when they are contextually relevant to the selection. The context menu doesn't show if there are no commands in it. Avoid showing a command that would show an error.


Cut
Copy
Lorem amet, consectetur adipiscing sit amet ante id justo euismod vehicula. Pellentesque egestas, enim sed gravida tempor, mi turpis ultrices est, ac
Editable text, nothing on the clipboard
Cut
Copy
Paste
Lorem amet, consectetur adipiscing sit amet ante id justo euismod vehicula. Pellentesque egestas, enim sed gravida tempor, mi turpis ultrices est, ac
Editable text, something on the clipboard

- **Command ordering**
  Add commands to the context menu in order of importance. Provide clipboard commands in the standard order at the bottom of the menu.


1: Bold
2: Highlight
3: Cut
4: Copy
5: Paste
Lorem ip onsectetur adipiscin ante id justo euismod vehicula. Pellentesque egestas, enim sed gravida tempor, mi turpis ultrices est, ac

- **5 Command limit**
  The context menu has a 5 command limit. Ask yourself the following questions if you are struggling to fit commands in the menu.
  - Does this need to be a UI command, can it be represented with a direct manipulation?
  - Do you need this command, what would your experience be like without it? Are you reducing concepts in increase confidence?
  - Is there another way to do this already? What is gained by duplicating in the context menu?
  - Does the item that this acts on have a 1up view? If so the command could be in the app bar on the 1up view instead on the context menu in the collection.
  - Does this always need to be shown, or only in certain contexts?

4.19.8.1.2 INAPPROPRIATE USE OF CONTEXT MENUS

- **Don't use the context menu when direct manipulation or selection is possible**

  The primary way to command applications is through direct maniputíon or selection and an app bar. When possible use an app bar or an app bar with selection rather than using a context menu to act on items.

- **Don't use the context menu when you it isn't required**

  If there is already a clear way to accomplish the scenario, do not create a context menu that creates a second way to complete that scenario. Trust that users will find commands by selecting an item or navigating to an item to act on it.

4.19.8.2 SAMPLE JAVASCRIPT CODE

Basic Menu in JavaScript Example:

This example creates a menu and sets up event handlers for when commands are selected.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <style type="text/css">
      #kitty {
        width: 160px;
        height: 120px;
        margin: 200px;
      }
    </style>
    <script type="text/javascript">
      function showMenu(image) {
        var menu = new Windows.UI.Immersive.PopupMenu();

        // Use the convenience overloaded constructor.
        menu.commands.append(new Windows.UI.Immersive.UICommand("Pet",
onPet));

        // Or the default constructor and set the properties yourself.
        var feed = new Windows.UI.Immersive.UICommand();
        feed.label = "Feed";
        feed.invoked = onFeed;
        feed.id = "FeedID"; // Can be anything
        menu.commands.append(feed);

        menu.showAsync().operation.start();
      }
      function onPet() {
        // Pet the cute cat.
      }
      function onFeed() {
        // Give her some food.
      }
    </script>
  </head>
  <body>
    <img id="kitty"
src="http://tailsmagazines.files.wordpress.com/2009/04/kitten.jpg"
oncontextmenu="showMenu(this); return false;" />
  </body>
</html>
```

4.19.8.3 REFERENCES

1. Context Menu API Doc
2. Context Menu Functional Spec

4.20 PERSONALITY

The personality, or look and feel of Windows, can be translated into your application by incorporating three key visual elements:

- Layout: Clean layouts that emphasize content over chrome
- Typography: Typography rich UI that helps delineate hierarchy
- Animations: Responsive and cohesive animations

Balancing these foundational elements with aspects of your application's brand, color, content, and interaction model will not only make your application look great, but will make it feel like cohesive member of the Windows 8 application family and make it familiar to users.

The following sections describe how applications manifest the look and feel of Windows 8 to create a single, cohesive experience across both the shell and individual applications.

4.20.1 ANIMATIONS

The Personality Visual Library (PVL) is an animation store that provides ***predesigned*** storyboards for common UI scenarios, such as:

- Navigation & Content Refresh
    - Loading: Loading your application
    - Page Transitions: Switching between two different views or pages
    - Content Transitions: Refreshing content within a page
    - Pagination: Navigating to new pages for reading scenarios
    - Crossfade: Refreshing the state of an element

- Collection Manipulation
    - Adds & Deletes: Showing new items, hiding deleted items, and reposition existing items.
    - Search: Adding and deleting items for search filtering
    - Selection: Selecting items in a collection to take action on them
    - Drag and Drop: Rearranging and moving elements
- Chrome Interaction
    - Showing custom App Bars: bring an off screen item into view, or slide a displayed item out of view.
    - Showing custom panels:
- User interaction
    - Tap & click: Showing user feedback for tap and click events
- Informative UI
    - Flyouts: Showing pop up messages that require user interaction
    - Inline messaging: Showing or hiding more information for an item by expanding and collapsing surrounding items.
    - Temporary UI: Fading and out UI that is temporarily not visible
- + Others…

Using the PVL animations will be the easiest way for developers to quickly add animations into their UI because it removes the need to design and tweak the animations from scratch for common application scenarios. Because of this, the PVL is not meant to be easily customizable or be used as a building block to create more complex animations. These animations are only intended to be used in the scenarios specified.

At the core, the PVL is the definition store with predefined:

- Transforms
- Timing functions
- And Storyboards

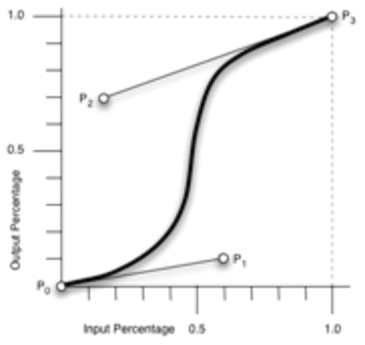
Each animation storyboard may contain a combination of these elements as well as information from the caller:

1. **Transforms**
   PVL transform animations enable you to animate an object's:

- Opacity
- Translation (2D)
- Rotation
- Scale (2D)
- Clip (2D)

The use and combinations of these transforms are predefined in an animations storyboard and should not be customized.

2. **Timing functions**
   PVL describes curves using splines (cubic Bezier functions) that use 4 control points, as show in the following illustration:



Timing functions are also predefined per animation should not be customized.

3. **Targets**
   A target is the object or sets of elements that the animation is applied to. Depending on your scenario, this may can be one or multiple elements.

For example, AddToGrid is an animation that can be used for adding an item to custom collection controls. This animation takes two targets:

- Target: The element(s) that is being added
- Affected: The element(s) that needs to be reposition / moved because of the added item.

The caller is responsible for identifying the target for the animation.

4. **Storyboarding**
   For each target, there is a set of transforms to be performed at specific times; these timed actions form a storyboard. The PVL specifies the begin time and duration of each transform. In some animations if the target supports a collection of items, there can be a staggered timing between each item in the target collection.

For example, for the AddToGrid animation, the neighboring elements to be repositioned can be staggered so the 1st starts moving at time 0ms, the 2nd starts at 5ms, the 3rd at 10ms, and so on, with a cap of 500ms at which point all remaining items move together.

The storyboard is predefined per animation and should not be customized.

5. **Static assets**
   There can be a standard theme file that should be used in conjunction with certain animations. For example, a "focus" animation (used when a control receives keyboard focus) might include a specific border around the control. PVL provides a reference to the asset files, which are stored in the system in the same way that theme files are normally stored in a UX Theme. These files can be retrieved via the existing Theming system's API (for example, GetThemeBitmap)

The definition store in and of itself is agnostic to the animation or composition platform. So PVL can be used as long as there is a component that can comprehend the transforms and can run the storyboards according to the timing functions. To consume the PVL, it is important to note that:

- The client must know visual elements' beginning state and end state (such as size and position). The client needs to manage content layout (PVL itself is agnostic to reflow logic).
- The client needs to render the animation, or use another compositor/animation engine to do so.

Currently, the PVL animations are exposed in two different ways, one for Jupiter clients and another for HTML clients.

4.20.2 WWAS AND ANIMATIONS

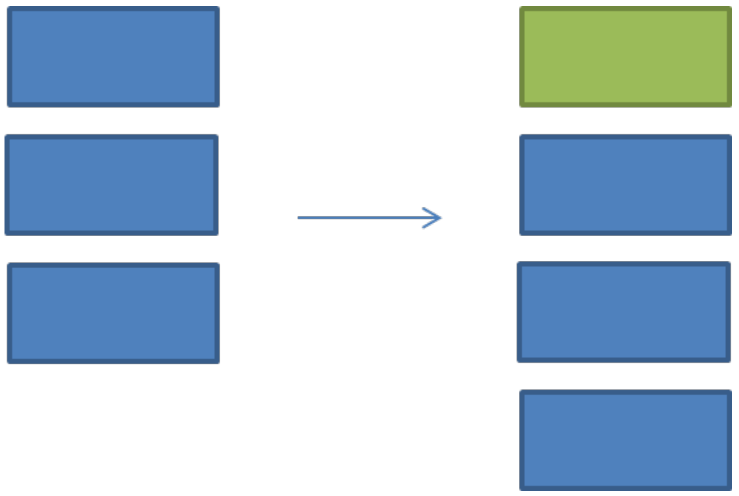
For HTML + JavaScript web apps, the animations are provided through a collection of JavaScript functions that map to the storyboards in the PVL definition store. These functions follow CSS3 drafts for transform, transition and animations. These functions are part of the animations.js file (which will be part of the SDK and can be included by 3rd party web apps), currently exposed under the WinJS.UI.Animation namespace.

The majority of functions accept an element id (or an array of element ids) as well as an optional event handler for when the animation has completed and, in some cases, other parameters. The PVL JavaScript sets up a CSS transition/animation and initiates the animation; no further action is required on the part of the caller in most cases. When the animation is complete, the PVL cleans up the transition properties of the affected objects.

The following is a simple example for using the "fade out" animation, which changes the opacity at a specific predefined rate:

```
// Win JS Files(ex: animations.js) must be referenced
Win.UI.Animation.fadeOut(document.getElementById("img1"));
```

Here a more involved example is the "addToList" animation, which adds an item to a list, while moving its neighbors to make room for the new item.



The addToList animation takes two parameters:

- The element that is being added.
- An array of affected elements that will have to be translated to make way for the new element.

```
// Win JS Files(ex: animations.js) must be referenced

var add = Win.UI.Animation.createAddToListAnimation(added, affected);

// Put elements into their desired post-animation states (in this case,
// insert new element into DOM tree, change position of affected
// objects if necessary)

add.execute(); // Run the animation
```

This example uses the addToList animation in a web app:

```
// Win JS Files(ex: animations.js) must be referenced

function insertObjectAtTopOfList(list) {
        var newItem = document.createElement("div");
        newItem.className = "listItem";

        var addToList = Win.UI.Animation.createAddToListAnimation(newItem,
list.children);

        list.insertBefore(newItem, list.firstChild);

        addToList.execute();
}
```

4.21 KEYBOARD / INPUT PANE

The Input Panel enables users to enter input directly on the screen while in the new immersive experiences of Windows 8. However, since the Input Panel will occlude applications, developers need a way to listen to changes in the state of the Input Panel. They should be able to quickly and easily understand how much of the window is obscured, so that they can fit their content into the available space. Supported Frameworks will provide a default experience, but developers can customize that experience at will. The default experience will ensure that the input field that caused the Input Panel to invoke is scrolled into view, though the details of that implementation are framework specific.

The Input Panel APIs enable the application to listen for changes in the state of the Input Pane and indicate whether the system should also handle the Input Pane event.

4.21.1 IHM SCENARIOS

The following are the different scenarios that can occur based on the type of handlers that are available when the input panel is invoked:

1. No App Handler, No Host Handler
   - Developer Ignores the InputPanelEvents
   - Host ignores the InputPanelEvents
   - Input Pane is shown, overlays application window.

2. No App Handler, Host Handler exists (This is the 90% case – the developer doesn't have to do any work, and the app host handles the Input pane)
   - Developer ignores the InputPanelEvents
   - Input Pane is shown
   - Host does their default behavior
     - For WWA host and WinRT UI, this behavior is to change the viewport without triggering a new layout

3. App Handler present
   - Developer sets up an event handler for the WinRT InputPane event
   - InputPane is shown
   - Host ignores the event
   - The app reacts as specified by the developer.

   It's worth noting that the developer can use the Host's reaction, and then add on additional behaviors.

4.21.2 SAMPLE JAVASCRIPT CODE

Reacting to input pane showing/hiding/hidden events:

```
// Reacting to the Input Panel Showing

Windows.UI.Immersive.InputPane.getForCurrentView().addEventListener("showi
ng", onInputPaneShowing);

Windows.UI.Immersive.InputPane.getForCurrentView().addEventListener("hidin
g", onInputPaneHiding);

function onInputPaneShowing(e)
{
    var occludedRect = e.occludedRect;

    // For this hypothetical application, the developer decided that 400
pixels is
    // the minimum height that will work for her current layout. When the
    // app gets the InputPaneShowing message, the pane is beginning to
animate in.

    if (occludedRect.Top < 400)
    {
        // In this scenario, the developer decides to remove some elements
(perhaps
        // a fixed navbar) and dim the screen to give focus to the text
element
        var elementsToRemove =
document.getElementsByName("extraneousElements");

        // The app developer will not use default framework animation
        // Note that the IHM uses the Win.UI.Animations.hidePanel and
        // Win.UI.Animations.showPanel animations for its timing curves
        _StartElementRemovalAnimations(elementsToRemove);
        _StartScreenDimAnimation();
    }

    // This developer does not want the framework's focused element
visibility
    // code/animation to clash with his/her own
    e.ensuredFocusedElementInView = true;
}

function onInputPaneHiding(e)
{
    // In this case, the Input Pane is leaving. The developer can use
    // this message to start animations.
    if (_ExtraElementsWereRemoved())
    {
        _StartElementAdditionAnimations();
    }

    // This developer does not want the framework's focused element
visibility
    // code/animation to clash with his/her own
    e.ensuredFocusedElementInView = true;
}
```

Using default animations in cases where you cannot do your own animation or handle the event due to cross-boundary security issues:

```
function OnLoad()
{

Windows.UI.Immersive.InputPane.getForCurrentView().addEventListener("showi
ng", onInputPaneShowing);

Windows.UI.Immersive.InputPane.getForCurrentView().addEventListener("hidin
g", onInputPaneHiding);
}

var IsFacebookIframeFocused = false;

function onInputPaneShowing(e)
{
    // This hypothetical app hosts a facebook iframe
    // This app is not able to manipulate anything within the iframe, so
    // it will depend on the framework to do its default animation work
    IsFacebookIframeFocused = (document.activeElement.id ==
"FacebookIFrame");
    if (!IsFacebookIframeFocused)
    {
        _DoCustomApplicationAnimation();

        // This developer does not want the framework's focused element
visibility
        // code/animation to clash with his/her own
        e.ensuredFocusedElementInView = true;
    }
}

function onInputPaneHiding(e)
{
    If (!IsFacebookIframeFocused)
    {
        _DoCustomApplicationAnimationAgain();
        e.ensuredFocusedElementInView = true;
    }
}
```

4.21.3 REFERENCES

1. API spec
2. M3 Dev spec
3. M3 Functional spec

4.22 FILE ASSOCIATIONS AND PROTOCOL EXTENSIONS

4.22.1 FILE ASSOCIATIONS

Files with a common file name extension (.doc, .html, and so on) are of the same *type*. When developing an application that uses files, you can use existing file types. For example, if you create a new text editor, you might use the existing .txt file type. In other cases, you might need to create a new file type. In either case, you can provide the necessary information about file associations in the extensions portion of the application manifest. The following example registers an application as a handler for the ".foo" file type.

```
<Extension Category="windows.fileTypeAssociation">
   <FileTypeAssociation Name="foo">
      <DisplayName>Foo File</DisplayName>
      <Logo>RESOURCES\fooLogo.png</Logo>
      <InfoTip>A foo file</InfoTip>
      <EditFlags OpenIsSafe="true" AlwaysUnsafe="false" />
      <SupportedFileTypes>
         <FileType MediaType="images/png">.foo</FileType>
      </SupportedFileTypes>
    </FileTypeAssociation>
</Extension>
```

4.22.1.1 SYSTEM FLOW FOR FILE ASSOCIATIONS

1. A modern application that declares one or more file associations is deployed.
2. The application is added to the OpenWith list for its declared file types.
3. The user opens a file by double-clicking or selecting **Open** from the context menu.
4. If the user has selected the application as their default the app is activated by the Modern Activation process.
5. The application receives an activation event with a contract ID of **Windows.File**.
6. The application uses the activation eventArgs to obtain the vector of **IStorageFile** objects that are to be opened by the application.

The following example handles activation for the **Windows.File** contract:

```
<script type="text/javascript">

function activatedHandler(eventArgs)
{
    if (eventArgs.contractId == "Windows.File")
   {
          var fileList = eventArgs.files;
```

```
            //act on the file(s) populated in the fileList var.
    }
}

Windows.UI.WebUI.WebUIApplication.addEventListener("activated",
activatedHandler, false);

</script>
```

4.22.2 PROTOCOL EXTENSIONS

Applications can utilize protocol extensions to provide functionality that is activated by a URI (for example, mailto:foo@bar.com). When developing an application that uses protocols, you can use existing, well-defined protocols or you can create your own. In either case, you can provide the necessary information about protocols in the extensions portion of the application manifest. The following example registers an application as a handler of the mailto protocol.

```
<Extension Category="windows.protocol">
     <Protocol Name="mailto" />
</Extension>
```

4.22.2.1 SYSTEM FLOW FOR PROTOCOL EXTENSIONS

1. A modern application that declares one or more protocol registrations is deployed.
2. The application is added to the Open With list for its declared protocol schemes.
3. The user clicks on a protocol link, such as mailto:foo@bar.com.
4. If the user has selected the application as their default the app is activated by the Modern Application Activation process.
5. The application receives an activation event with a contract ID of **Windows.Protocol**.
6. The application uses the activation eventArgs object to obtain the full protocol URI.

The following example handles activation for the **Windows.Protocol** contract:

```
<script type="text/javascript">

function activatedHandler(eventArgs)
{
       if (eventArgs.contractId == "Windows.Protocol")
       {
          var uri = eventArgs.uri;
          //act on the protocol populated in the protocol var.
       }
}

Windows.UI.WebUI.WebUIApplication.addEventListener("activated",
activatedHandler, false);
```

```
</script>
```

4.22.3 LAUNCHING FILES OR PROTOCOLS

Modern applications can invoke the default handler for a particular file or URI. For example, a modern mail app can receive file attachments and pass them to the proper handler when the user attempts to open them. To do this, applications must use the new WinRT Launcher API. This API enables applications to hand off files or URIs to another application, which opens in its proper security context. Existing methods for accomplishing this task, such as **ShellExecute**, do not work in a modern app because they do not launch the handler application in the correct security context.

4.22.3.1 SYSTEM FLOW FOR LAUNCHING FILES OR PROTOCOLS

1. A modern application creates an object, **StorageFile**, or URI to pass to another app.
2. The application UI presents the intent to launch the object to the end user.
3. The user chooses to open the object.
4. The application calls the **LaunchDefaultProgramForFile** or **LaunchDefaultProgram** method on the **Windows.UI.Activation.Launcher** API
5. The default handler for the file or URI is launched and passed the object during activation.

The following example invokes a file handler:

```
function LaunchFileHandler(file)
{
   Windows.ApplicationModel.Launcher.launchDefaultProgramForFile(file);
}
```

The next example invokes a URI handler:

```
function LaunchProtocolHandler(uri)
{
   Windows.ApplicationModel.Launcher.launchDefaultProgram(uri);
}
```

4.22.4 WINRT API DETAILS

Windows.UI.Activation.FileActivationContext

Methods	**Description**
Files	Returns the vector of **StorageFile** objects that were passed to the application during activation.

Windows.UI.Activation.ProtocolActivationContext	
Methods	**Description**
Uri	Returns the **Uri** object that was passed to the application during activation.

Windows.UI.Activation.Launcher	
Methods	**Description**
InvokeFileHandler	Passes the specified **StorageFile** to the application that is the default application for files of that type.
InvokeUriHandler	Passes the specified **Uri** to the application that is the default application for that protocol.

4.22.5 REFERENCES

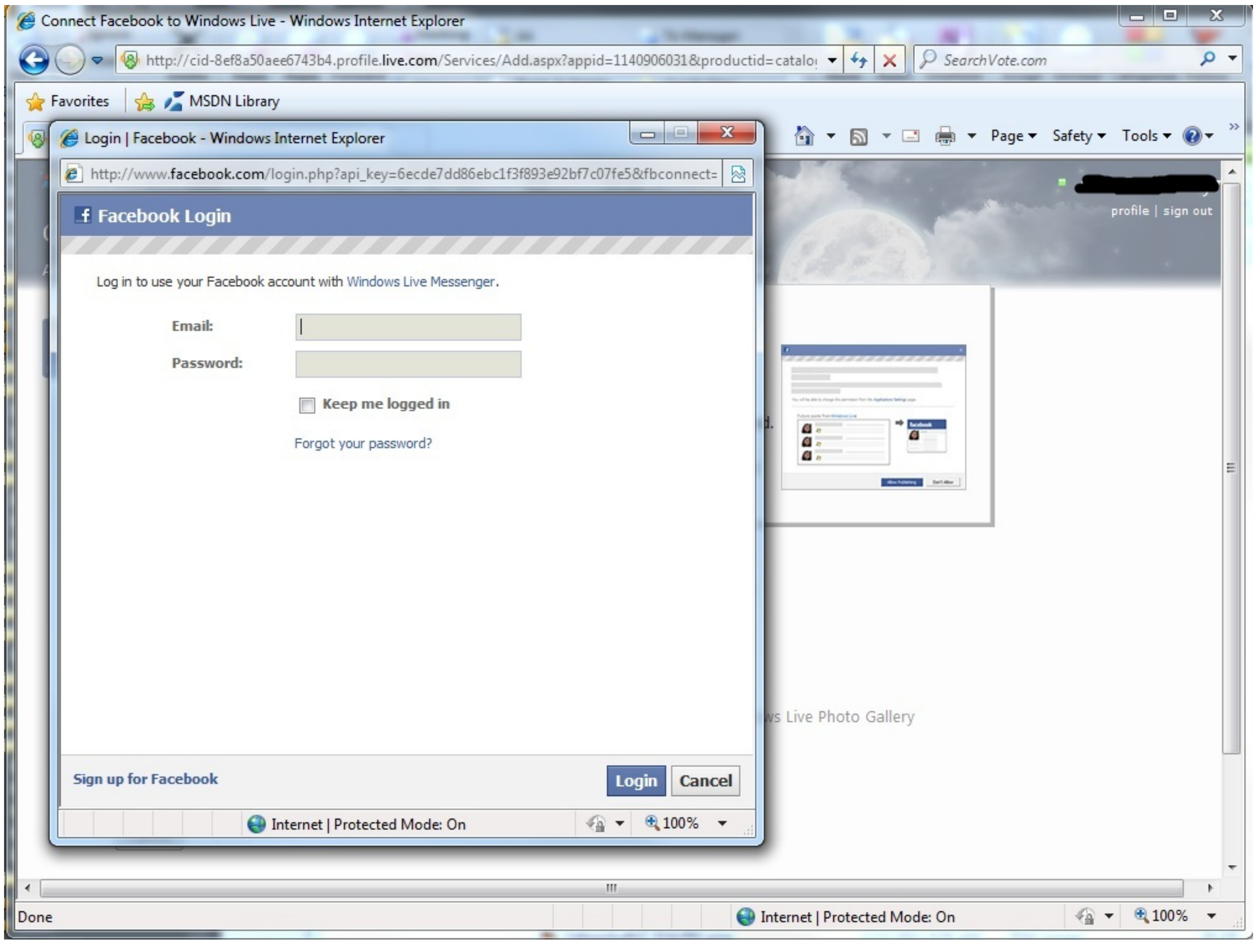
1. Trust Association Launching Functional Spec
2. Trust Association Launching Dev Design Spec
3. Trust Association Launching API Summary Doc
4. Samples at: \\appx-share\public\aandrejs

4.23 CREDENTIAL MANAGEMENT

With the proliferation of online services, credentials are playing an ever larger role in the typical web user's experience; users manage an ever growing set of username and password combinations. Predictably, users are responding to this by choosing a single password or small set of passwords that they use across all their online services.

4.23.1 WEB SERVICE AUTHENTICATION

The Web Authentication Broker APIs allow modern applications to authenticate and get an authorization before they can connect to Internet services like Live, Facebook, and Twitter using protocols such as OAuth 2.0. Usually this involves having a user to logon to the online service provider and go through an authorization step. The following are example of a typical user flow in the browser (here a user is connecting his Live account to Facebook).


Connect Facebook to Windows Live - Windows Internet Explorer

http://cid-8ef8a50aee6743b4.profile.live.com/Services/Add.aspx?appid=1140906031&productid=catalo

Login | Facebook - Windows Internet Explorer

http://www.facebook.com/login.php?api_key=6ecde7dd86ebc1f3f893e92bf7c07fe5&fbconnect=

Facebook Login

Log in to use your Facebook account with Windows Live Messenger.

Email:

Password:

Keep me logged in

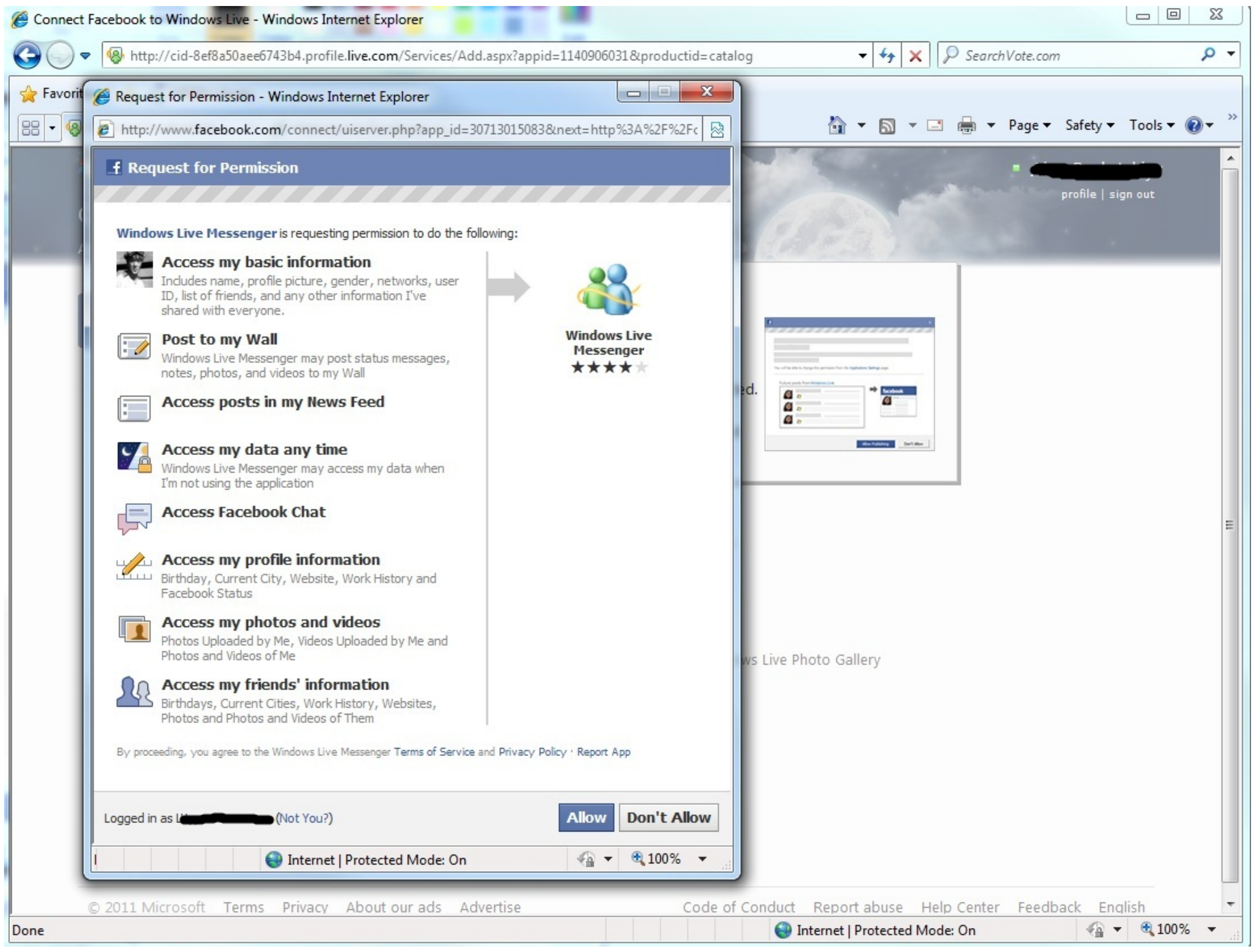
Forgot your password?

Sign up for Facebook

Login Cancel

Internet | Protected Mode: On 100%

profile | sign out [redacted]


Connect Facebook to Windows Live - Windows Internet Explorer

http://cid-8ef8a50aee6743b4.profile.live.com/Services/Add.aspx?appid=1140906031&productid=catalog

Request for Permission - Windows Internet Explorer

http://www.facebook.com/connect/uiserver.php?app_id=30713015083&next=http%3A%2F%2Fc

Request for Permission

Windows Live Messenger is requesting permission to do the following:

Access my basic information
Includes name, profile picture, gender, networks, user ID, list of friends, and any other information I've shared with everyone.

Post to my Wall
Windows Live Messenger may post status messages, notes, photos, and videos to my Wall

Access posts in my News Feed

Access my data any time
Windows Live Messenger may access my data when I'm not using the application

Access Facebook Chat

Access my profile information
Birthday, Current City, Website, Work History and Facebook Status

Access my photos and videos
Photos Uploaded by Me, Videos Uploaded by Me and Photos and Videos of Me

Access my friends' information
Birthdays, Current Cities, Work History, Websites, Photos and Photos and Videos of Them

Windows Live Messenger ★★★★

By proceeding, you agree to the Windows Live Messenger Terms of Service and Privacy Policy · Report App

Logged in as [redacted] (Not You?)

Allow | Don't Allow

Internet | Protected Mode: On 100%

[redacted]
profile | sign out

© 2011 Microsoft Terms Privacy About our ads Advertise Code of Conduct Report abuse Help Center Feedback English

Done Internet | Protected Mode: On 100%

In the desktop world applications have to use browser controls to implement this flow. In the browser world a JavaScript popup is most often used.

In Windows 8 the Web Auth APIs provide an easy way for application to complete such flows by exposing a method that lets applications specify a starting URL and a terminating URL. The API then creates a modern dialog over the application window that displays web pages (starting URL is the first page that gets displayed) necessary to complete the task. When a terminating URL is reached, the modern dialog is taken down and the API returns the authentication/authorization data (usually the last URL + query strings, fragments, etc) back to the application. Note that the web pages are hosted in the process outside of the application process to keep user credentials isolated from the calling app.

4.23.1.1 WINRT API DETAILS

Warning: The following API syntax is for the PDC-4 build. It will be updated for the PDC-3 build to support new async call pattern, global WinRT namespace reorganization, as well as new functionality to support single sign on scenarios. See references to links to updated API documentation.

Class WebAuthenticationBroker : IWebAuthenticationBrokerStatics

Method/Property	Description
AuthenticateAsync	Creates a WebAuthenticationOperation that is then used to complete the web authentication and get the results back. The method takes two parameters, a request URL and a callback URL. The request URL is the URL for the first web page to render and contains protocol parameters required by the online service provider. The callback URL is a terminator. The web auth broker matches (left substring) all URLs that it is about to navigate to and if the URLs match, the navigation is terminated. Note that last URL is never navigated to.

Class WebAuthenticationOperation : IWebAuthenticationOperation

Method/Property	Description
Completed	Gets/Sets the event hander for the operation completion.
GetResults	Returns the WebAuthenticationResult object that has the results of the async operation.

Class WebAuthenticationResult : IWebAuthenticationResult

Method/Property	Description
ResponseData	Returns the URL that the web auth broker matched with the callback URL including query string and fragment portions. Application should use this URL to extract the protocol data such as authorization token from it according to the guidance provided by the online service provider.
ResponseStatus	Returns the status with which the operation has completed. There are distinct values for user canceling the operation as well as for HTTP protocol specific errors. For all other Win32 errors an exception is thrown.
ResponseErrorDetail	Used to obtain the HTTP error code when ResponseStatus is set to ErrorHttp.

4.23.1.2 SAMPLE JAVASCRIPT CODE

```
//this function is called when a user clicks on the Facebook button within
a modern app in order to authentication to Facebook and access photos
stored in Facebook's site

function appUserClickOnFaceBookButton()
{

    var webAuthenticationOperation =
Windows.Security.Authentication.WebAuthenticationBroker.AuthenticateAsync(
    new
Uri("https://www.facebook.com/login.php?api_key=123&display=popup"),
//request Uri
    new Uri("http://login.kodak.com/"));  //callback Uri


    webAuthenticationOperation.then(
            function (webAuthenticationOperation)
            {

                    var webAuthenticationResult =
webAuthenticationOperation.GetResults();

                    if (webAuthenticationResult.ResponseStatus !=
Windows.Security.Authentication.WebAuthenticationStatus.Success)
                            return;

                    var responseData =
webAuthenticationResult.ResponseData;

                    //app now can obtain the accessToken from responseData.

            }
        );
}
```

4.23.1.3 REFERENCES

1. Quick Start Guide in Java Script
2. Functional spec
3. Functional spec
4. Dev spec
5. API spec (PDC-4)
6. API Spec (PDC-3 and later)

4.23.2 CREDVAULT

The credential management feature provides a common way to store and manage passcodes, passphrases, and other identification in a safe and secure store. This secure storage mechanism, known as the Windows Credential Value or CredVault, makes using passwords and user names easy—users only need to remember their Windows login information to login to any site.

Since credentials management is an important and sensitive task, the Windows Credential Vault provides the means to securely store such important information and exposes a simple, safe set of WinRT APIs for accessing this information.

4.23.2.1 SAMPLE JAVASCRIPT CODE

Credential Management:

```
//Initialization
var vault = new Windows.Security.Credentials.PasswordVault();

       var cred = new Windows.Security.Credentials.PasswordCredential();
       cred.Resource = "example.com";
       cred.Username = "user";
       cred.Password = "password";

       var cred = new
Windows.Security.Credentials.PasswordCredential("example.com", "user",
"password");

//Writing credentials
vault.Add(cred);
vault.Add(new
Windows.Security.Credentials.PasswordCredential("example.com", "user",
"password"));

//Searching for credentials
var findCreds = vault.FindAllByResource("example.com");
var creds = vault.GetAll();

//Reading credentials
var cred = vault.Get("example.com", "user");

//Deleting credentials
vault.Remove(cred);

//Using the credential
       if (cred.Password == null)
              cred.GetPassword();
       alert("username: " + cred.Username + " and password: " +
cred.Password);
```

4.23.3 REFERENCES

1. Credential Management Dev Spec

4.24 SUMMARY OF MODERN APPLICATION EXTENSION POINTS

Contract	Required	Manifest Registration	Flow Location	Activation Required
Tiles	Yes	Yes	MoGo	Yes
Secondary Tiles	No	No	MoGo	Yes
Tile Badges	No	Yes	MoGo	No

Lock Screen Notifications	No	Yes	Lock Screen	No
Toasts	No	Yes	MoBody Overlay	Yes
Push Notifications	No	No	MoGo	Yes
Snapping	Yes	No	MoBar	No
Suspend/Resume (PLM)	Yes	Background tasks only	App Code	No
Search	No	Yes	Charms Bar	Yes
Share	No	Yes, Targets Only	Charms Bar	Targets Only
Settings	No	No	Charms Bar	No
File Picker	No	No	File Picker	No
File Picker Extension	No	Yes	File Picker	Yes
File System Broker	No	Yes – for accessing libraries	App Code	No
Connect-Print	No	No	Charms Bar	No
Connect-PlayTo	No	No	Charms Bar	No
Connect-Send	No	Yes, Targets Only	Charms Bar	Targets Only
Splash Screen	Yes	Yes	Any app-instance launch	Shows on app activation
Keyboard / Input Pane	No	No	App Code	No
File Associations and Protocol Extensions	No	Yes, Targets Only	Clicking File or Protocol link	Targets Only
Credential Management	No	No	App Code	No

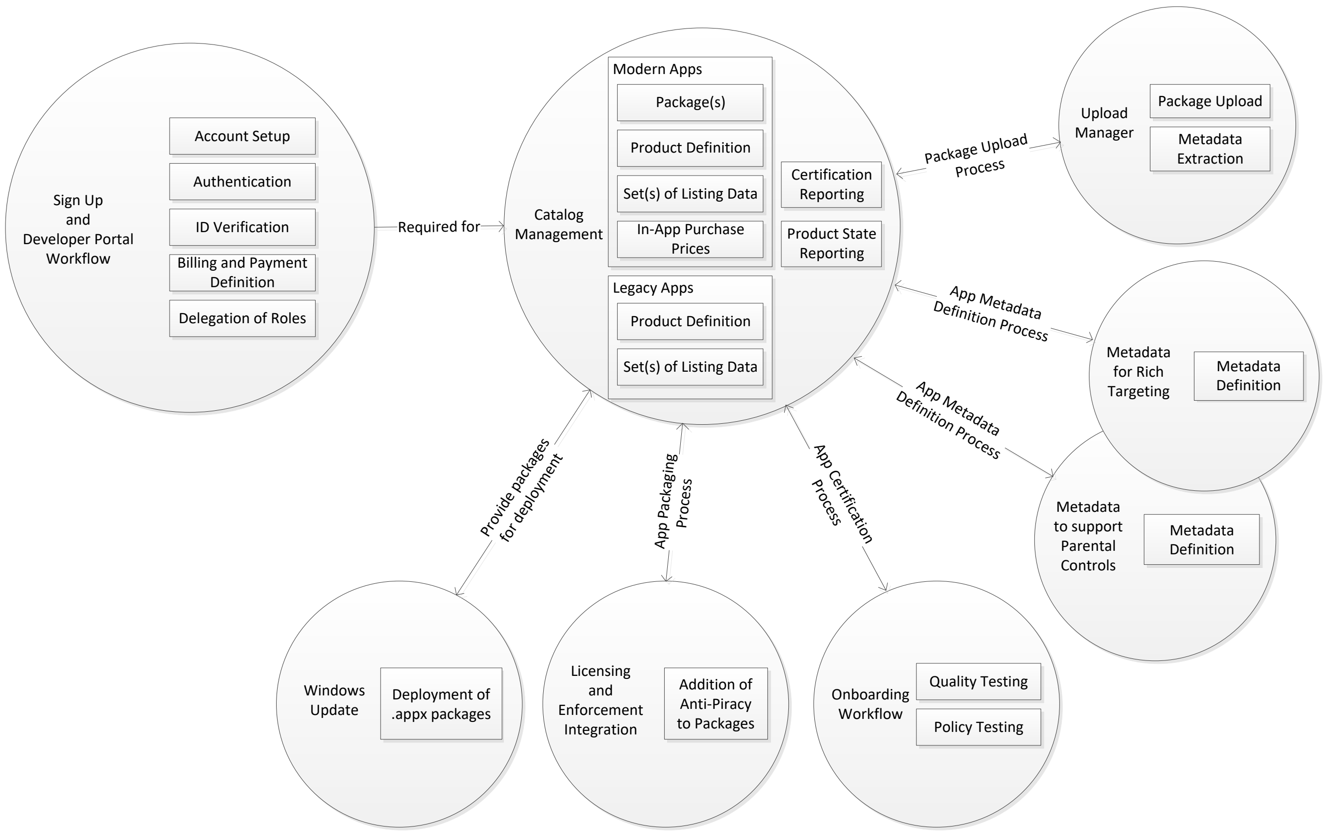
5 APP STORE AND APPLICATION ACQUISITION

The App Store is where consumers discover and acquire the best-in-class modern Windows applications for all their needs. As a result, the Store is the best way for developers and publishers to monetize their offerings and works best with the Windows 8 modern application platform.

5.1 THE ONBOARDING AND CERTIFICATION EXPERIENCE

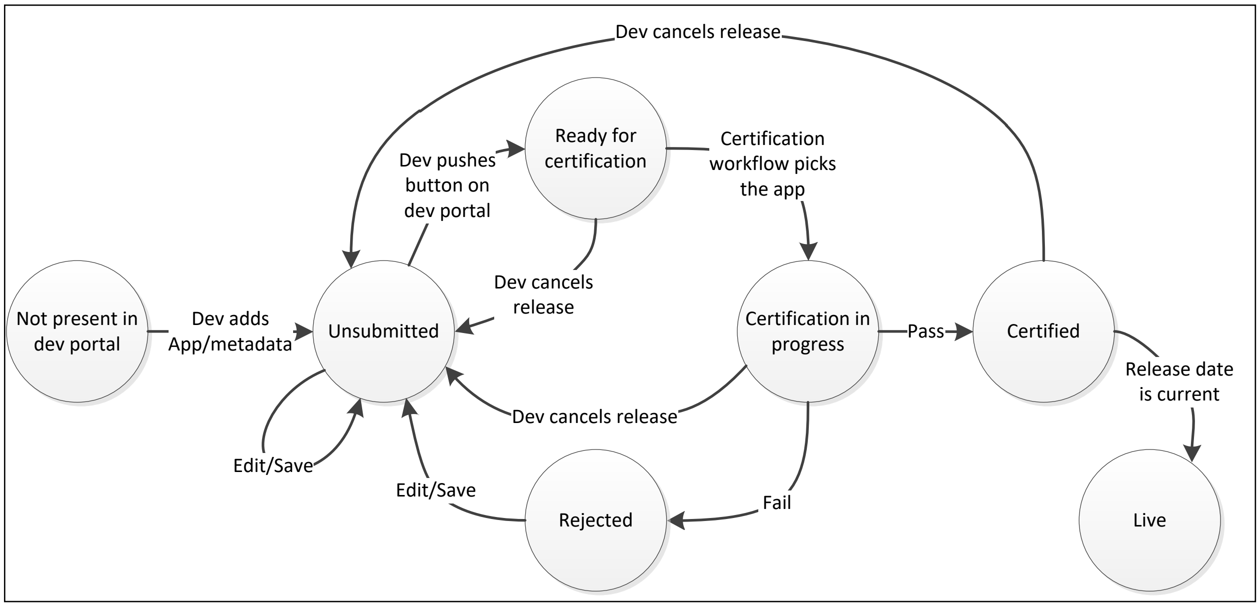
For each product sold through the store, developers will provide key information which will enable users to make informed purchase decisions. This includes user visible information about the offering such as descriptive information, screenshots, and price. The upload of the product as well as supplying additional marketing information occurs through the Developer Portal. Developers will also be able to manage their application offerings throughout the offering life-cycle. You can add, update, and remove offerings from the App Store catalog and determine the status of an application that is being processed for approval.

5.1.1 DEVELOPER PORTAL HIGH LEVEL DIAGRAM



Applications submitted to the store go through quality checks before being made available to customers. The certification process involves automated and manual testing of the application as well as static validation to verify that it meets all the policies and requirements of the App Store.

5.1.2 CERTIFICATION WORKFLOW HIGH LEVEL DIAGRAM



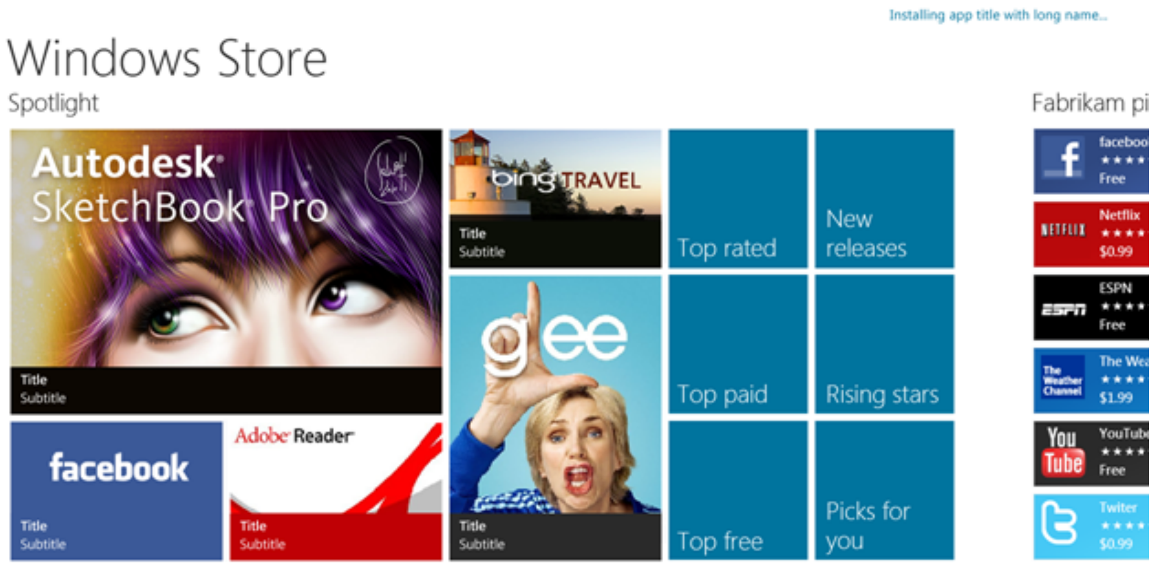
5.1.3 REFERENCES

1. App Store Developer Policies // update link

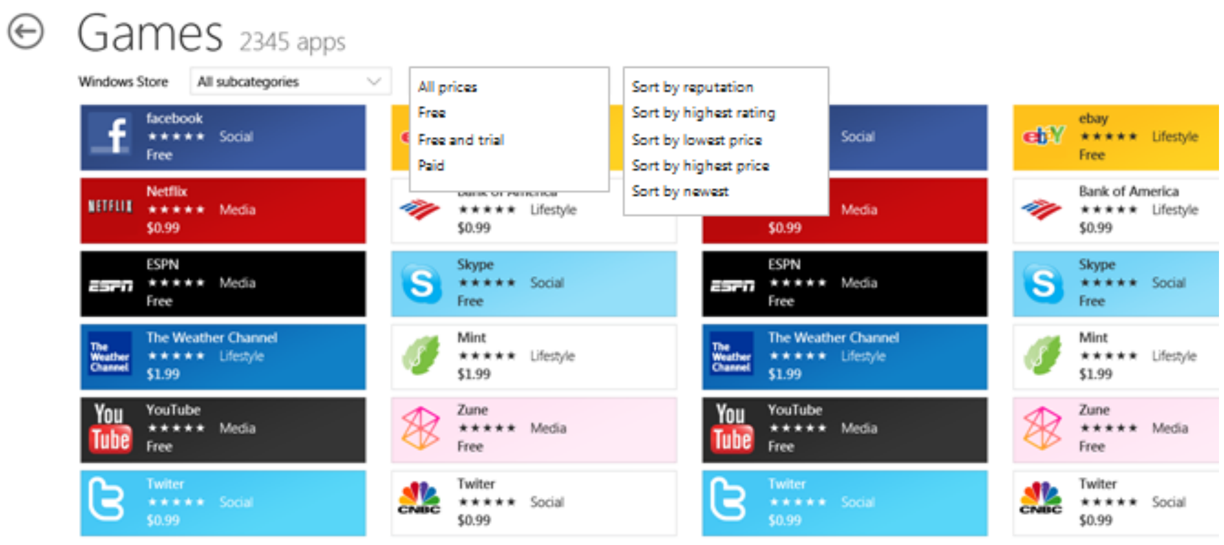
5.2 DISCOVERY POINTS: BROWSE, SEARCH, LISTS, AND DEEP LINKING

The App Store client consumer experience provides several ways for users to find and discover applications to purchase. Users may search and browse by categories and subcategories and can filter and sort by price, rating, release date, and so on.

The App Store Homepage:



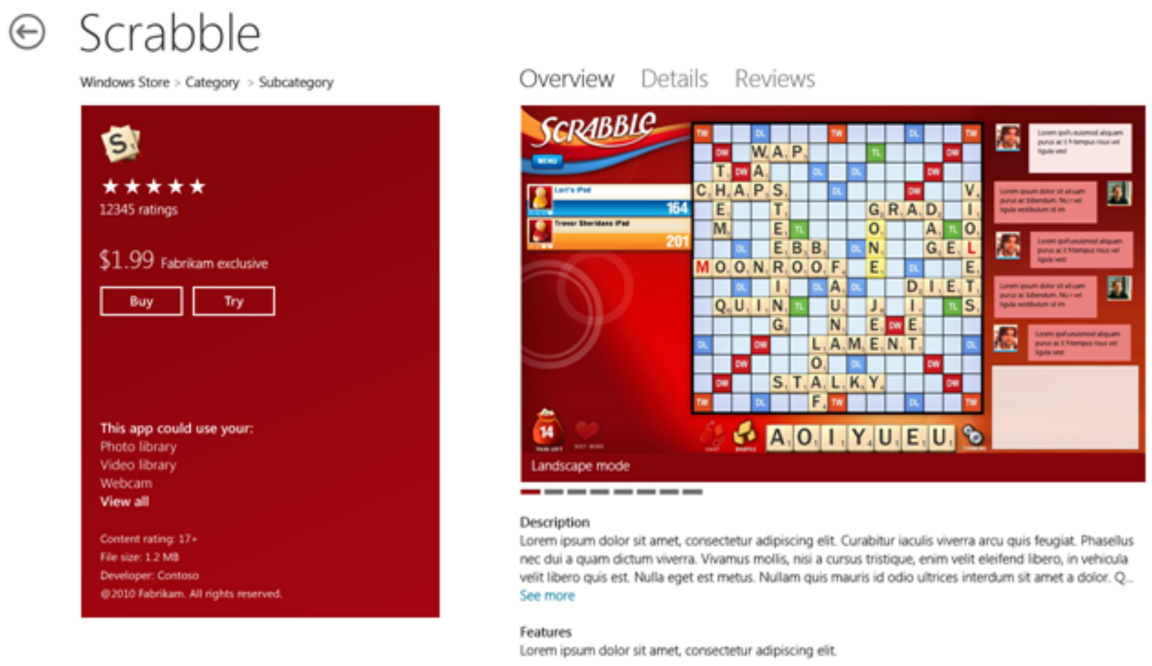
Browsing the App Store:


Games 2345 apps
Windows Store
All subcategories
All prices
Free
Free and trial
Paid
Sort by reputation
Sort by highest rating
Sort by lowest price
Sort by highest price
Sort by newest
facebook ★★★★★ Social Free
Netflix ★★★★★ Media $0.99
ESPN ★★★★★ Media Free
The Weather Channel ★★★★★ Lifestyle $1.99
YouTube ★★★★★ Media Free
Twitter ★★★★★ Social $0.99
Bank of America ★★★★★ Lifestyle $0.99
Skype ★★★★★ Social Free
Mint ★★★★★ Lifestyle $1.99
Zune ★★★★★ Media Free
Twitter ★★★★★ Social $0.99
Social
Media $0.99
ESPN ★★★★★ Media Free
The Weather Channel ★★★★★ Lifestyle $1.99
YouTube ★★★★★ Media Free
Twitter ★★★★★ Social $0.99
ebay ★★★★★ Lifestyle Free
Bank of America ★★★★★ Lifestyle $0.99
Skype ★★★★★ Social Free
Mint ★★★★★ Lifestyle $1.99
Zune ★★★★★ Media Free
Twitter ★★★★★ Social $0.99

There are landing pages that contain editorial/featured content and data-generated content (such as Most Popular Applications and Applications Recommended for You [based on purchasing history]). Applications with a higher reputation will have a higher probability of being featured on a landing page.

5.3 SELLING THE APP: PRODUCT DESCRIPTION PAGE AND ACQUISITION

A product description page contains the key information about an application, and it's where customers learn to use, acquire, and rate and review the application. The data on this page is provided primarily by ISVs via metadata extracted from the package and provided explicitly at onboarding time, but also contains data from users in the form of ratings and reviews. Developers use this page to market their applications to customers.



Scrabble

Windows Store > Category > Subcategory

★★★★★
12345 ratings

$1.99 Fabrikam exclusive

Buy | Try

This app could use your:
Photo library
Video library
Webcam
View all

Content rating: 17+
File size: 1.2 MB
Developer: Contoso
©2010 Fabrikam. All rights reserved.

Overview Details Reviews

Landscape mode

Description
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur iaculis viverra arcu quis feugiat. Phasellus nec dui a quam dictum viverra. Vivamus mollis, nisi a cursus tristique, enim velit eleifend libero, in vehicula velit libero quis est. Nulla eget est metus. Nullam quis mauris id odio ultrices interdum sit amet a dolor. Q...
See more

Features
Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Figure 8: An product description page

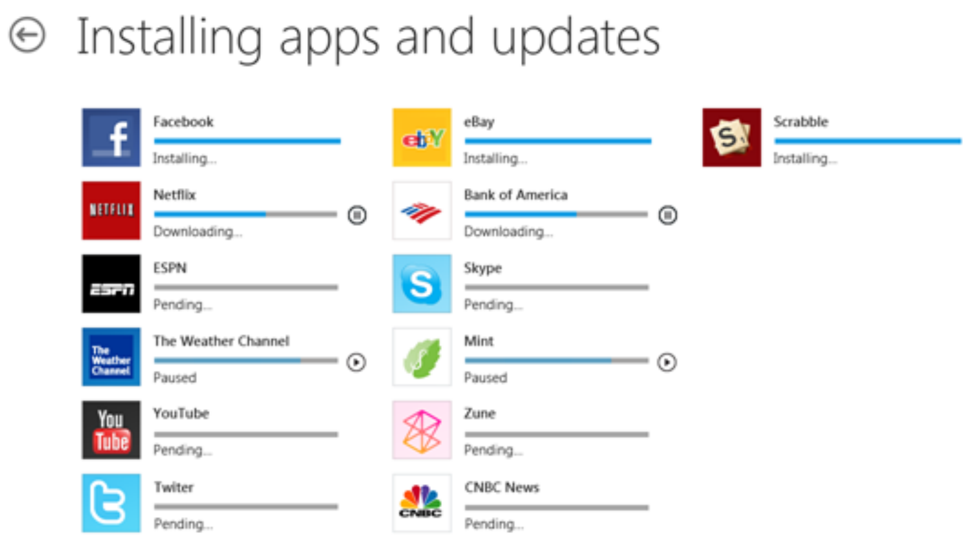
How acquisition works:

1. The user presses button to try or buy a product.
2. The store client displays confirmation UI for paid apps (trials and free apps do not require confirmation) and the user confirms the purchase.
3. Re-authentication may occur for paid application purchase depending on user settings.
4. The payment is processed and license is downloaded to the client.
5. Application installation starts automatically; progress can be seen in the store Installs page.

5.4 INSTALL UX

Once a user initiates acquisition of the application, the user will remain in the store and is taken back to their previous page in order to allow the user to continue shopping for applications. If a user wishes to view detailed installation progress, there is a page in the store to do so.

App install and update progress:



Users can pause a download in progress by tapping the tile. This pauses the download until the user taps the tile again as a toggle.

5.5 LICENSING AND THE IN-APP PURCHASING EXPERIENCE

The store provides a mechanism for applications to check the state of a user's license for the application. There are two primary uses for the licensing: providing trials and enabling in-app purchases.

5.5.1 TRIAL TO FULL LICENSE CONVERSION

The App Store allows developers to provide time-limited trials for the products they sell. The developer can opt to make it a simple time-limited trial or to restrict functionality during this trial period. The user can convert their license from a trial to a full license within the store client user experience. Application developers can also begin the upgrade from within the application itself.

How in-app trial conversion works:

1. The application triggers the in-app conversion flow through one of the following mechanisms:
    a. The application provides a call to action, which the user selects.
    b. The application triggers the conversion flow based on an event, such as the trial licensing expiring.
2. The user is presented with App Store user interface to confirm their purchase.
3. Re-authentication occurs if user has not authenticated within a 5-minute window.
4. The payment is processed and license is modified.
5. An updated license is returned to application through a callback.

5.5.1.1 LIMIT APP FUNCTIONALITY DURING TRIALS

A developer wants to enable a trial but prevent the user from “saving” documents during the trial to encourage them to purchase the full version. The developer includes a “save” button in the HTML:

```
<input type="button" id="save" />
```

He creates a function which runs when the page is done loading which disables the “save” button if there’s only a trial license:

```
// save a reference to LicenseInformation static class
var currentProduct = Windows.ApplicationModel.Store.CurrentProduct;
var licenseInformation = currentProduct.LicenseInformation;

// when the page is done loading, check for trial license
document.onreadystatechange = initializePage;
function initializePage() {
    if (document.readyState == "complete") {
        // check license
        checkLicense();
    }
}

function checkLicense() {
    // disable the button if only running with a trial license
    var saveBtn = document.getElementById('save');
    if (saveBtn != null)
        saveBtn.disabled = licenseInformation.isTrial;
}
```

5.5.1.2 PURCHASE FULL LICENSE WHEN TRIAL EXPIRES

The developer wants the app to be alerted if the trial expires while running so he can offer the user to upgrade to a full license. He does several things:

1. He registers a handler for the `LicenseChanged` event in the `initializePage` function.

1. He expands the license check code to check for expiration.
2. Adds a function to offer the customer a chance to pay for a full license.

```
// save a reference to LicenseInformation static class
var currentProduct = Windows.ApplicationModel.Store.CurrentProduct;
var licenseInformation = currentProduct.LicenseInformation;

// when the page is done loading, check for trial license
document.onreadystatechange = initializePage;
function initializePage() {
    if (document.readyState == "complete") {
        // check license
        checkLicense();
        // register a function for license status changed notifications
        licenseInformation.addEventListener("LicenseChanged",
checkLicense);
    }
}

// checks the status of the trial license and takes action if it's now
Expired
function checkLicense() {
    // disable the button if only running with a trial license
    var saveBtn = document.getElementById('save');
    if (saveBtn != null){
        saveBtn.disabled = licenseInformation.isTrial;
    }
    // if trial license has expired, navigate to a trial expired page
    if (!licenseInformation.isActive){
        document.location = "trialexpired.html";
    }
}

// button or link on trialexpired page calls this method to buy full
version of the application
function trySellUpgrade() {
    // launch the purchase UX
    var requestPurchaseOperation =
currentProduct.requestProductPurchaseAsync();
    requestPurchaseOperation.completed = completePurchaseRequest;
    requestPurchaseOperation.start();
}

// handle the purchase request completion
function completePurchaseRequest(requestPurchaseOperation) {
    if (requestPurchaseOperation.status == AsyncStatus.completed) {
        requestPurchaseOperation.close();
        if (licenseInformation.isActive) {
            // show that the purchase succeeded here…
            ...
        }
    }
    else if (requestPurchaseOperation.status == AsyncStatus.cancelled ||
            requestPurchaseOperation.status == AsyncStatus.error) {
        // show that purchase did not succeed here…
        ...
    }
}
```

5.5.2 IN-APP PURCHASES

Applications themselves can sell items, such as addition content, levels, or virtual goods. The store models these purchases as modifications to the user's application license.

How in-app purchases work:

1. The application triggers the in-app purchase flow through one of the following mechanisms:
    a. The application provides a call to action, which the user selects.
    b. The application triggers the conversion flow based on an event.
2. The user is presented with App Store user interface to confirm their purchase
3. Re-authentication occurs if user has not authenticated within the past 5 minutes.
4. The payment is processed and license is modified.
5. An updated license is returned to callback.

5.5.2.1 COMMERCE SNIPPETS – FREE APP SELLS EXTRA FEATURE

A developer wants to provide a feature which requires the user to pay extra to use it. They define a feature purchase on the developer web portal and then add code to their app to unlock the feature with a purchase that's launched from within their application.

Their app might have a button that launches the feature as follows:

```
<input type="button" id="extraFeature" />
```

This code checks for the feature's license whenever the user attempts to use the feature and if it doesn't exist, it attempts to sell the license to the user:

```
// save a reference to LicenseInformation static class
var currentProduct = Windows.ApplicationModel.Store.CurrentProduct;
var licenseInformation = currentProduct.LicenseInformation;

// the string value assigned in the App Store dev portal for purchasing
the new feature
var extraFeatureID = "Feature 1";

// prompts user to buy feature
function trySellFeature() {
    // give them the option to buy the feature
    var confirmResult = confirm("Would you like to purchase this
feature?");
    if (confirmResult == true) {
        // if they want to purchase the feature, launch the store purchase
UX to confirm the purchase
        var requestPurchaseOperation =
currentProduct.requestFeaturePurchaseAsync(extraFeatureID);
        requestPurchaseOperation.completed = completePurchaseRequest;
        requestPurchaseOperation.start();
    }
}

// handle the purchase request completion
function completePurchaseRequest(requestPurchaseOperation) {
    if (requestPurchaseOperation.status == AsyncStatus.completed) {
        requestPurchaseOperation.close();
        if (licenseInformation.isActive) {
            // show feature UX
            document.location = "extrafeature.html";
        }
    }
    else if (requestPurchaseOperation.status == AsyncStatus.cancelled ||
             requestPurchaseOperation.status == AsyncStatus.error) {
        // show that purchase did not succeed here…
        alert("The purchase did not succeed");
    }
}

// launches the extra feature
function launchExtraFeature() {
    // check if they have an active license for the feature
    if
(licenseInformation.featureLicenses.lookup(extraFeatureID).isActive) {
        // show feature UX
        document.location = "extrafeature.html";
    } else {
        // give user option to buy extra feature if they haven't already
        trySellFeature();
        // doesn’t navigate away – user is left in same context to try
launching
        // feature again after modal purchase operation finishes
    }
}

// when page is done loading, finish initializing the application
document.onreadystatechange = initialize;
function initialize() {
    if (document.readyState == "complete") {
        // register click handler for launching extra feature
        var launchFeatureBtn = document.getElementById('extraFeature');
        if(launchFeatureBtn != null)
            launchFeatureBtn.onclick = launchExtraFeature;
    }
}
```

The example above just launches the App Store purchase UX to sell the feature. If the developer wants to show information about the item(s) before launching the store's purchase UX, they can get information about it through us – we get it indirectly from our web services. They can show the name and price for the feature using another button:

```
<input type="button" id="sell_button" />
```

To do this, they first make changes to the initialize function to register the button's event handler and call the load listing information function:

```
// when page is done loading, finish initializing the application
document.onreadystatechange = initialize;
function initialize() {
    if (document.readyState == "complete") {
        // register click handler for launching extra feature
        launchFeatureBtn = document.getElementById('extraFeature');
        if(launchFeatureBtn != null)
            launchFeatureBtn.onclick = launchExtraFeature;

        // register click handler to sell extra feature
        sellFeatureBtn = document.getElementById('sell_button');
        if (sellFeatureBtn != null) {
            sellFeatureBtn.onclick = trySellFeature;
            sellFeatureBtn.disabled = true;
        }

        // load the listing metadata information to advertise the
feature's name and price
        loadListingInformation();
    }
}

// reference to the sell button
var sellFeatureBtn;
```

Then add this code, which loads the listing information, to the feature purchasing sample above:

```
// load the listing metadata information for this application
function loadListingInformation() {
    var loadListingOperation =
currentProduct.loadListingInformationAsync();
    // callback function to update UX once listing metadata information
has completed loading
    loadListingOperation.completed = loadListingCompleted;
    loadListingOperation.start();
}

// process the completion of the listing metadata load operation
function loadListingCompleted(loadListingOperation) {
    if (loadListingOperation.status == AsyncStatus.completed) {
        // if the operation completed, load the listing object
        var prodListing = loadListingOperation.getResult();
        // set the button's label to show the name and price of the 
Feature and enable the button
        if (sellFeatureBtn != null) {
            sellFeatureBtn.value = 
prodListing.FeatureListings.lookup(extraFeatureID).name + ": " + 
prodListing.FeatureListings.lookup(extraFeatureID).formattedPrice;
            sellFeatureBtn.disabled = false;
        }
        // close the async operation object
        loadListingOperation.close()
    }
}
```

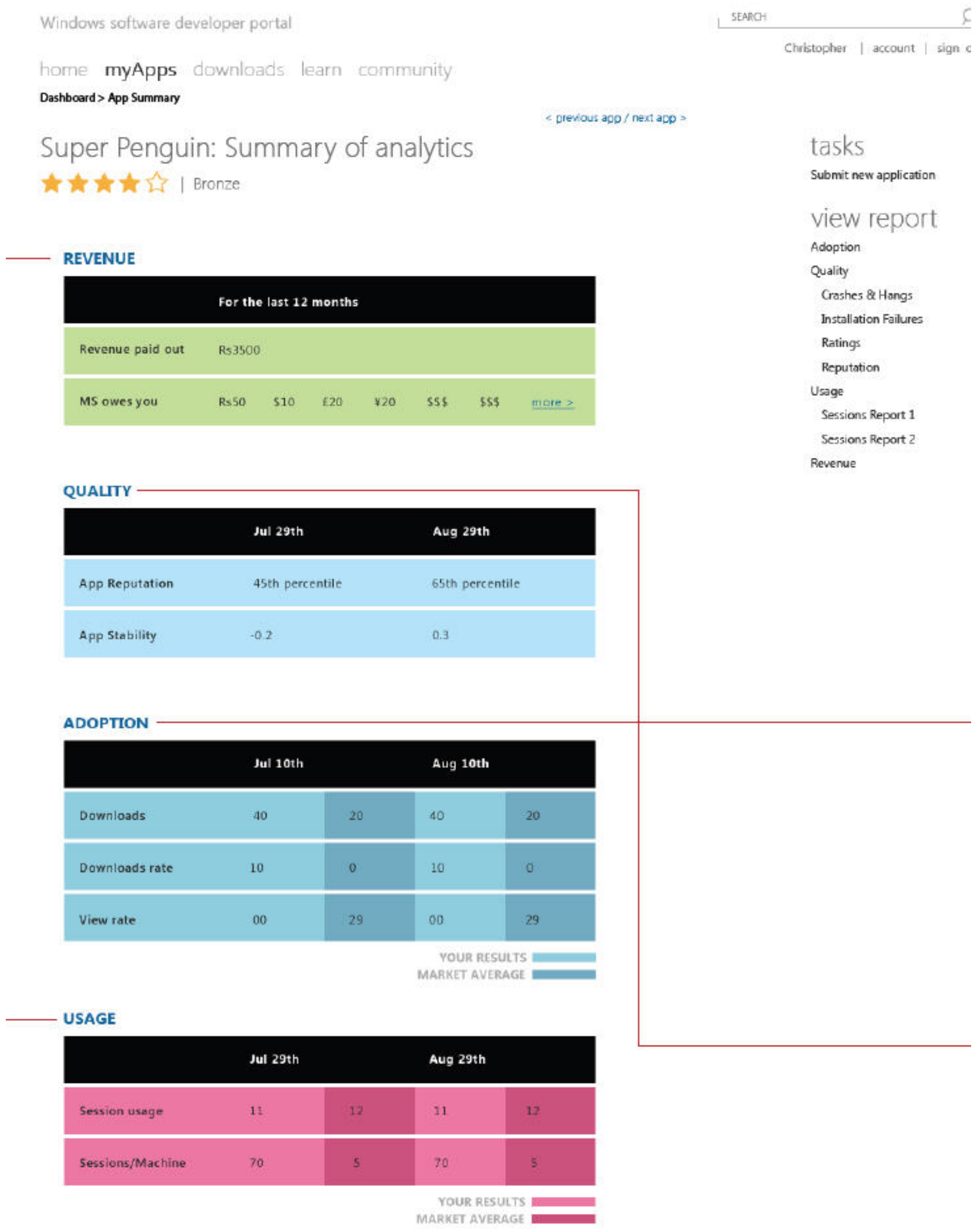
5.6 SERVICING & UPDATING THE APP

The App Store provides a mechanism for developers to deliver updates to their products. To submit an update, you perform the same steps as when you upload a new product. When you update the product, you can also make changes to the product metadata. Once an update is released, the product is updated on client machines that have the product installed. Users must perform manual updates. Manual updates require the user to initiate updates from within the store client. Updates may be pre-downloaded opportunistically to make the act of updating fast and fluid.

5.7 ANALYTICS & GETTING PAID

The Developer Portal provides detailed analytics about the product. The portal provides a number of reports, including adoption, quality, usage, revenue, and market data mining reports.

The following illustration shows a sample reports page.



Windows software developer portal

SEARCH

Christopher | account | sign c

home myApps downloads learn community

Dashboard > App Summary

< previous app / next app >

Super Penguin: Summary of analytics

★★★★☆ | Bronze

tasks

Submit new application

view report

Adoption
Quality
Crashes & Hangs
Installation Failures
Ratings
Reputation
Usage
Sessions Report 1
Sessions Report 2
Revenue

REVENUE

	For the last 12 months						
Revenue paid out	Rs3500						
MS owes you	Rs50	$10	£20	¥20	$$$	$$$	more >

QUALITY

	Jul 29th	Aug 29th
App Reputation	45th percentile	65th percentile
App Stability	-0.2	0.3

ADOPTION

	Jul 10th		Aug 10th	
Downloads	40	20	40	20
Downloads rate	10	0	10	0
View rate	00	29	00	29

YOUR RESULTS ▬
MARKET AVERAGE ▬

USAGE

	Jul 29th		Aug 29th	
Session usage	11	12	11	12
Sessions/Machine	70	5	70	5

YOUR RESULTS ▬
MARKET AVERAGE ▬

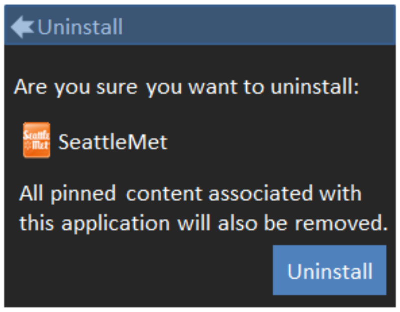
5.8 UNINSTALL UX

Products are uninstalled from application tiles in MoGo. The uninstall user experience is designed to:

- Prevent accidental uninstalls.
- Provide enough information so that the user can make the right decision about uninstall.
- Be simple.

Uninstall is initiated by invoking the tile context menu (by pressing and tapping or right-clicking the tile), and selecting **Uninstall**. This uninstalls all applications in the product. It is not possible to uninstall individual applications within a product. Any pinned content is also removed.

When uninstall is initiated, a flyout is shown to let the user confirm the uninstall operation.


Uninstall
Are you sure you want to uninstall:
SeattleMet
All pinned content associated with this application will also be removed.
Uninstall

As soon as the user confirms uninstall, the tiles are immediately removed.

6 DEVELOPER TOOLS AND DEBUGGING

The tooling and debugging experience of the modern application developer consists of four major stages:

1. Search and discovery
2. Acquisition of tools
3. Application development
4. Debugging

6.1 SEARCH AND DISCOVERY

The debugging and tools story begins with effective search and discovery. The first result returned by search engines includes the Developer Center, which contains all of the information necessary for building modern apps. The Developer Center provides a school-like learning experience; a series of modules, or lessons, are available for developers to work through, learning at their own pace. Technical documentation on MSDN supports these lessons, and functions as the Developer Center library.

Additionally, forums will be hosted, providing a central location for Windows 8 developers to request and provide peer guidance.

The Developer Center also functions as the central repository for application templates and sample code that you can copy, paste, and modify for your applications. Many of these samples and templates will be provided by Windows, but the Developer Center will also highlight the best 3rd-party developer template and sample submissions.

This sample code follows Windows coding guidelines; using it can simplify the development process and make it easier to follow Windows guidelines and best practices.

The Windows Developer Center will guide developers to the next stage of the process, Acquisition of Tools, by prominently linking to both Visual Studio Express, and the full Windows SDK. Either of these downloads provides everything a developer needs to create, test and publish Modern Applications.

6.2 ACQUISITION OF TOOLS

6.2.1 VISUAL STUDIO

Visual Studio Express is a full-featured IDE that enables Modern Application development in any supported programming language. The Visual Studio tools handle tasks such as manifest creation and validation, generation of WWA metadata, pre-generated accessible markup, and localization support. The IDE will provide control objects, such as radio buttons, in the drag-and-drop designer in order to further simplify application development.

Visual Studio express supports Windows Web Applications and native applications (C/C++, C#, VB).

To simplify the development experience, Visual Studio Express will:

- Build .APPX packages.
- Automatically generate localization indices.
- Provide a GUI Package Manifest Designer that includes:
    a. Support for controlling Application UI (primarily Tile content)
    b. Support for adding capabilities to applications.
    c. Support for adding contracts to applications.
    d. Support for adding Content URIs
    e. Options to control application packaging.
- Support multiple layout development (docked, portrait, landscape, full screen, and so on).
- Enable developers to debug modern applications running in an immersive window.
    a. RDP loopback session to allow developers to see their application running while debugging

   b. DWM window to provide a snapshot of the application while debugging (targeted at high fidelity graphics)
- Provide C# and C++ developers with public symbols they can provide to the Windows Store to enhance the provided developer analytics
- Provide Windows Store validation at development time to provide developers with confidence that their application will be accepted following submission

Visual Studio will ship with the Modern SDK, a simplified version of the Windows SDK specifically designed for Modern Application development. The Modern SDK only contains tools that enable key developer scenarios, and it only contains APIs that have been approved for use by the App Store.

The following tools will be included in the Modern SDK (not guaranteed to be either a final or exhaustive list):

- MakeCert.exe: A tool used to generate signing certificates.
- CertMgr.exe: A tool used to manage signing certificates on a machine.
- SignTool.exe: A tool used to sign files with a signing certificate.
- MakePRI.exe: A tool used to localize an application.
- WSLK: A tool used to verify applications before submission to the App Store.

The limitations on the Modern SDK make it difficult to write applications that will not pass App Store onboarding, and minimizes the total size of Visual Studio Express.

6.2.2 EXPRESSION BLEND

In addition to Visual Studio, developers are free to use Expression Blend to build their applications. While Visual Studio primarily targets developers and coding-intensive projects, Expression Blend targets designers and makes it easy to build great looking Modern Applications. While you can write and test code from Blend, its primary offering is a WYSIWYG design surface that has been engineered to provide a high fidelity representation of what an application will look like, and how it will perform, when completed and installed by an end user.

The WYSIWYG design surface is able to deliver a compelling experience by running the application unbeknownst to the developer. This allows for innovative features, including real time application interaction and rendering of the HTML/CSS/JS elements so that developers and designers can see the true behavior of the application while they design it.

Expression Blend will come in two flavors; one is targeted at WWAs, while the other is optimized for MCAs.

6.2.3 ADDITIONAL CHOICES

While Visual Studio will be the recommended development environment, other environments are supported. Using a text editor such as Vi or Notepad++ along with the Windows SDK and PowerShell cmdlets is the most lightweight option available, though clearly not full featured. 3rd parties, such as Adobe and Eclipse, will also be able to provide tools for Modern Application development.

The full Windows SDK will be available for these developers. This will allow 3rd party IDEs to create their own Modern Application development environment, as well as enabling the command line/test editor developers. All Modern Application tools shipped in the Modern SDK will be included in the full SDK, as well as:

- MakeAppx.exe: A tool used to create .appx packages

In addition to containing the tools necessary for Modern Application development, this SDK SKU will include desktop and driver utilities that are not appropriate for Modern Applications.

6.3 APPLICATION DEVELOPMENT

The Modern Application model has been designed for simplicity, but some constructs may initially appear obtuse to new developers. To help developers transition to the new model, our tools that contain accurate and helpful developer documentation. For any given feature, a variety of documentation exists, ranging from short, scenario-focused topics to whitepapers that describe the Modern Application model in great detail. This documentation will be designed to guide developer through the overall experience, with emphasis being given to core development, which breaks into a few distinct pieces:

- **Create and Edit**: The developer knows what they want to build, and begins implementing their design.
- **Compile, Build and Test**: The developer begins testing their application, validating functionality and improving the design.
- **Debug and Diagnose**: The developer has built major pieces of their application and is focused on diagnosing and fixing bugs in their program.
- **Package and Submit**: The developer is happy with their application and is ready to submit it to the Windows Store. They build a package and onboard it via the Developer Portal.

Once an application has been live on the Store, developer may also elect to go back and improve their application based upon feedback and analytics surface via the Store itself.

6.4 DEBUGGING

Debugging of modern applications is fully supported by Visual Studio, and multiple immersive window modes are available to ease development. In order to install and debug a Modern Application, a few things must happen.

- A Developer License must be installed on the machine
- The application must be fully deployed
- If the application is a WWA, switching the WWA host into debug mode
- Attaching the debugger

6.4.1 OBTAINING A DEVELOPER LICENSE

The App Store is the source for Modern Applications. This enables Microsoft to ensure an excellent user experience for each application acquired. A ramification of this, however, is that users and developers are not able to install Modern Applications from any source they choose. As a result, Windows supports developers installing uncertified applications while at the same time discouraging standard users from doing so.

To install an application which is not acquired from the App Store, you obtain a developer license by registering with the App Store. Payment is not required – only a Live ID.

Once you've registered with the App Store, you use a tool provided with the SDK (Modern or Full), which downloads a license from the Store (Note: Visual Studio and Expression Blend automate this for developers through public APIs). This license will enable a developer to install any application.

6.4.2 DEPLOYING THE APPLICATION

In Development Mode, application deployment occurs directly from the binary location of the app's source, known as "run from layout" deployment. A full registration of the application does occur, but the physical files continue to reside in the location they were registered from. To enhance system robustness, the Package Manifest is copied to the single-instance storage location in addition to the source location.

6.4.3 IMMERSIVE WINDOW

Due to the nature of immersive windowing in the MoSh, a debug environment is needed for developers to fully test and debug their applications. Since apps generally run full-screen, it would otherwise be difficult to have both the debugger and the immersive application displaying at the same time. In the debugging mode for the MoSh, a developer can:

- View their application and Visual Studio side by side
- Test the application in multiple views (immersive, docked, and background).
- Test for different resolutions.
- Test landscape and portrait layouts.
- Emulate the touch environment so that you can simulate touch interaction using your existing hardware.

- Emulate the sensor environment for GPS devices
- Emulate the app store for testing in-app purchases.
- Test the tile representation of the application and toast notifications.

6.4.4 (WWA ONLY) SWITCHING THE WWA HOST INTO DEBUG MODE

The JavaScript engine that powers Windows Web Applications is highly optimized and uses JITing to "compile" the otherwise interpreted JavaScript language for greater performance. While JITing provides better performance, it has the side effect of making JavaScript code more difficult to debug. When a developer signals that they'd like to debug a WWA, the IDE informs the WWA Host, which sets an internal flag that's read by the JavaScript engine. The JavaScript engine prepares the process for debugger attach by turning off JITing of JavaScript code and loading PDM.DLL (as shipped by VS or a 3rd party).

6.4.5 ATTACHING THE DEBUGGER

The final step in the debugging process is attaching the debugger. Upon release, the WWA Host and the MoSh return to their normal state.

7 DEVELOPER GUIDANCE

This section provides an overview of MoSh elements for developers who want to plan and write early MoSh apps.

7.1 ONE OR MULTIPLE APPS PER PACKAGE?

One of the decisions you face when building immersive applications is whether to put one application in a package by itself or many applications in a single package. When deciding, you should consider the business and technical impact of your deployment choice.

7.1.1 BUSINESS IMPACT

The following sections describe the business impacts of different packaging choices.

7.1.1.1 ONE PACKAGE, ONE PRODUCT IN THE APP STORE

Each product in the App Store maps to a single package. That means that you have one price and one product page for a package, regardless of how many tile-producing applications the package contains. The user sees a single listing in the Store for those applications. For example, Office might have a single package that contains Word, Excel, and PowerPoint. The user would have to find the "Office" application in the App Store and buy it to get these three applications. There would be no listings in the store for

Word, Excel, and PowerPoint individually (though keyword searches could still return the "Office" package).

7.1.1.2 ONE PRODUCT, ONE LICENSE

Building on the preceding example, if Word, Excel, PowerPoint are all available in the store as separate products (in individual packages) as well as a product called Office with all three applications (in its own individual package), buying one product won't give you a discount or credit towards the other products.

If the user were to buy all four of these products, they would not share state, share settings, or be able to integrate with each other because they each have their own isolated view of the system.

7.1.1.3 UPGRADES ARE ALWAYS FREE

An upgrade, no matter how major, is always free. The App Store is discussing whether or not developers could charge for an update to a new major version of the product, but this is currently not in the POR for Windows 8 RTM.

7.1.1.4 PRODUCTS CAN'T INSTALL OR UNINSTALL OTHER PRODUCTS

From a licensing and installation perspective, there is no relationship between products. That means that if you purchase Word as a separate product, you cannot buy or install PowerPoint or other parts of the Office suite from within the Word application.

File type associations let you launch the application through a link to a file if it is already installed, but there is no way to turn that file association into an application purchase.

You might be able to link to the App Store product page for the user to buy another application, but it is not possible to send users to the App Store when they try to open a particular file type, then automatically open the document after they've purchased the app that can open that file.

7.1.1.5 IN APP PURCHASES CAN "UNLOCK" APPLICATIONS BUT NOT APP TILES

There are APIs for in-application purchases that allow developers to define code that unlocks features based on some transaction with the user. These APIs do not enable the application to download and execute new code: the Application Container prohibits downloading and executing new code, as does App Store policy. Developers using in-app purchasing APIs to unlock features should download all of the bits as a part of the initial package.

There is no way to create an App Tile outside of deployment; you cannot add additional App Tiles when the user makes in-app purchases. For example, there is no way to only show the user a PowerPoint App Tile for the Office product they bought in the App Store and then, after an in-app purchase, another App Tile is for Word.

7.1.2 TECHNICAL IMPACT

This section focuses on the technical impact of deciding whether to break up your application into individual packages for each application or whether to bundle all of your applications into a single package. There are three main areas that impact this decision:

- Application Container
- App Store Technical Policy for Application Packages and Framework Packages
- Immersive Shell Contracts Scope

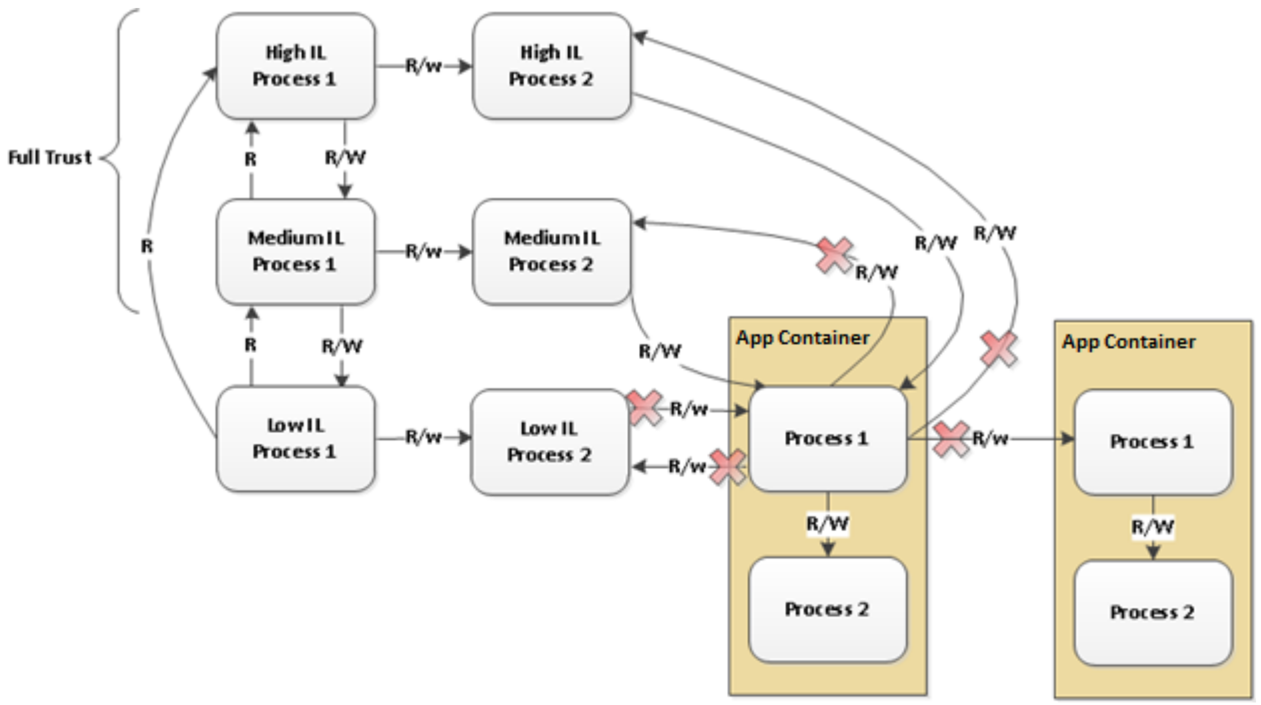
7.1.2.1 APPLICATION CONTAINER

All immersive applications install through the App Store and are constrained through the Application Container. The Application Container is an Access Control List (ACL) -based resource security mechanism that protects applications from each other and enables customers to install apps from the App Store with confidence. For more details about Application Container, please see the Application Container wiki.

How does the Application Container affect how you package your applications? If you choose to package your applications in separate packages, they can't do the following:

- Send messages back and forth.
- IPC.
- Share settings or storage.
- Share COM or MoCOM objects.
- Share code between packages.
- Know the file path to the other package.
- Share registry keys (or use the registry to communicate at all).
- Send network messages directly to each other.
- Write shared files to the user's profile.

Being in different packages completely isolates all of your applications from each other. The diagram that follows shows how the system can communicate with processes in Application Containers, but nothing else can. Being in the same package doesn't have the same restrictions. Multiple applications and multiple processes can be spawned within a single Application Container; applications in the same package are in the same Application Container and can therefore truly integrate with each other through means such as code sharing, IPC, and shared COM objects.



7.1.2.2 APP STORE POLICY LIMITATIONS FOR PACKAGES

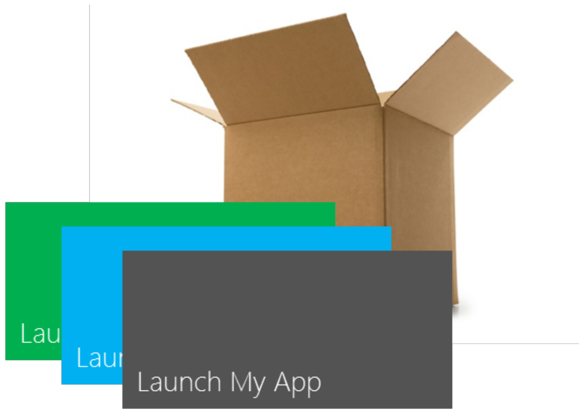
The DevX wiki provides partners with the latest information about how to think about packaging their applications. The presentation on package terminology is useful for understanding packaging as of M2. When determining whether to package your applications together or separately, there are some important technical App Store policy decisions to consider:

- **Application Packages** contain all of the resources, that when deployed, present a tailored application to users. One product in the App Store equals one application package, meaning:
    - Application developers cannot share code by creating a private package that is shared between multiple packaged applications.
    - Application developers must package all of their libraries, controls, resources (DLLs, languages, images, etc.) into a single application package. Using the Office example described previously, if Word and PowerPoint shared an MSO.DLL (10 MB), this DLL would need to be packaged separately in both the Word and PowerPoint packages if Word and PowerPoint were packaged separately, duplicating the size on disk and in memory. If Word and PowerPoint were in a single package, the DLL can be shared by both of them, reducing disk and memory usage.
    - Immersive applications will be prohibited from accessing APIs that would enable communicating with desktop applications and installed services that are not a part of

Windows that may be attempting to circumvent Application Container limitations. Therefore, creating integration between two immersive applications through installed desktop applications is prohibited by App Store policy.

- **Framework Packages** contain the runtimes and code resources that any Application Package can consume to build and render their tailored applications for users. There will only be a small selection of Framework Packages supported during the lifetime of Windows 8 by the App Store because of these frameworks cost much more to deploy and manage. All Framework Packages are 1st party (Microsoft-written code).
  - Because this Framework Package list is limited, developers needing to share code between a set of Application Packages will not be able to simply solve their problem by creating a Framework Package.
  - The App Store is focusing on "runtime" Framework Packages that support MoSh contracts and enable the creation of large sets of applications through new programming languages, controls, and APIs. Examples of runtime Framework Packages include the C Runtime and the Corsica JavaScript framework.

Application Packages **Framework Packages**



7.1.2.3 IMMERSIVE SHELL CONTRACTS

MoSh contracts can seem like they provide ways to work around some of the resource and communication limitations imposed by Application Container and the packaging limitations imposed through App Store policy. For example, the Sharing contract seems like a way for two applications, isolated by their Application Containers, to push data back and forth to each other. There are some problems with this approach:

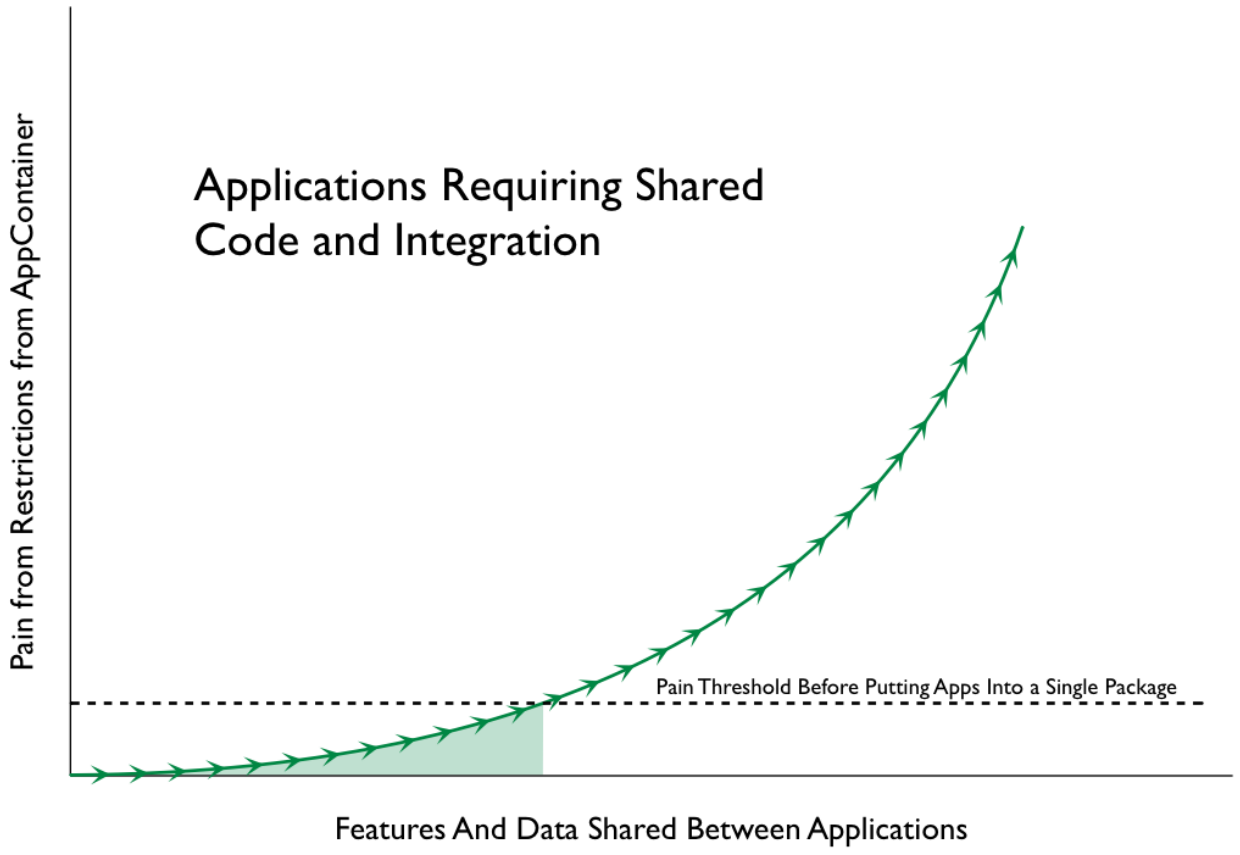
- Even though the Sharing contract does enable data to be passed between two applications, the data types are limited, and the cognitive disconnect the contract imposes (there is UI that pops up and must be interacted with to make a connection to a list of applications that may include your app and others) makes this solution suboptimal, especially for applications that require a high degree of integration.

- Share uses a push model where the user has completed their task in one application and is moving on to another application to finish the task completely. This doesn't work for tasks where the user expects to move quickly back and forth between the two applications.

7.1.3 THE SOLUTION

Circumventing Application Container restrictions by using network services and MoSh contracts is not a viable approach for most application. When the user will be switching back and forth between the applications, and when applications share settings and share objects in memory, you should consider packaging your applications into a single Application Package.

If your applications are very different and won't be used together, and if they do not share enough code to create disk space issues, separate your apps into multiple packages.



7.2 RELATIONSHIP BETWEEN MOSH AND DESKTOP APPS

MoSh apps are fundamentally different than classic, full-trust desktop apps. One mistake developers make when first considering how build a MoSh app is to think about porting existing desktop features or functionality into a new MoSh app.

The MoSh is an entirely new shell whose purpose, affordances, security model, and constraints are as different from the classic desktop app model as the DOS box is to the Windows desktop GUI.   Planning a MoSh app requires a new way of thinking about your users, scenarios, interaction model, data, storage, and resources.   The MoSh App First model also requires rethinking the relationship between users' content (files, pictures, videos, data) and their applications.

7.2.1 APP FIRST

The App First principle means that, by design, there is no "Windows Explorer" in the MoSh.   App First implies there is no way to natively browse or explore content outside of the framework of an App. Some apps might want to provide a content (or "library") view as their primary experience: for example, a media app might display a flat list of songs or albums, making it very easy for users to select and play music. Another example is a photo gallery app displays a flat or sorted list of users' locations and accompanying cloud-based photos so that users can casually browse or share their photos. However, there is no generic Windows-provided file-system "explorer" designed to help users launch apps through classic file associations.

7.2.2 FILE SYSTEM ACCESS VIA PICKERS

User-driven file system access is only possible through the MoSh Picker. The MoSh Picker enables users to quickly import or save files between their apps and common system locations and devices. The Picker is not intended to be a Common File Dialog. By design, there is no navigation pane, breadcrumb bar, or search. It is not intended for management tasks such as renaming, deleting, rotating, or managing files. It has one focused purpose: to enable users to quickly import or save files between their apps and the file system.

7.2.2.1 GUIDANCE FOR GALLERY APPS

Gallery apps are defined as apps that aspire to expose a large, aggregated data set in a browse-able view, letting you smoothly and seamlessly move throughout the set and take action on items. They are predominantly a grid that allows smooth panning, high fidelity thumbnails, sorting, filtering, resizing items in the view, launching, connecting to web services, and search. They also will increasingly be expected to work over multiple sources such as local libraries, HomeGroups, offline content (a detached USB drive or device), smart phones, web storage (such as SkyDrive), social networking, and sharing websites like Flickr or Facebook. (Of all these, only local libraries and HomeGroup are supported by Windows indexing.)

The biggest question for a gallery app is whether to create its own custom database or leverage the File Broker to store data, sync data, generate views, and search. For rising star developer apps, the platform solution should be sufficient, but most top tier competitive gallery apps will need create their own database.

The Windows platform solution is recommended for apps with the following characteristics:

- All data for the app lives in libraries or indexed scopes and only use system supported file types (jpeg, wma, mp3, xml, text, and Office document formats).
    - The file broker\indexer can provide views and content search over file system content that is in an indexed location and corresponds to a supported file type.
- The app does not need to extend the property system, provide per-user metadata, or extend metadata to non-supported types (such as tagging AVI or PNG files).

Creating a custom database, rather than using the Windows platform solution, is recommended for apps with the following characteristics:

- The app needs to represent data beyond default File system capabilities for supported types or use a new storage type that isn't supported in-box.
- The app needs to generate fast views over non-indexed locations, such as the web, UNC, thumb drives, other devices, and optical media.
- The app needs fine-tuned for better performance or needs granular control of its views.

7.2.2.2 WITHIN GALLERY APPS THERE ARE 2 DISTINCT TYPES OF APPS

There are two types of gallery apps: top-tier competitive gallery apps and rising star developer gallery apps.

7.2.2.2.1 TOP-TIER COMPETITIVE GALLERY APPS

The primary purpose of theses app is to expose a flat or aggregated gallery, often from multiple data sources. These apps are typically in highly competitive verticals such as music, pictures, or videos and compete against a multitude of apps vying for market share (for example, Zune, WMP, Live Photo Gallery, iTunes, iPhoto, Picasa, and Adobe elements). These apps typically have a large number of developers building and streamlining the app, and they will go to great lengths for any competitive edge in the space by adding enhanced features, service integration, and speed improvements. If the Windows platform does not provide an ideal solution aligning with their goals, these companies will invest in large dev teams to create and manage their own databases. Building and managing their own database means these developers will have more granular control over their views, performance, syncing, and aggregation without constraints imposed by using a more general-purpose system component, such as the Windows Indexer. Typically, these apps also have a monetization path beyond selling the application that supports the larger dev investment required to write and manage a database. Many of these apps will also be competing with their existing desktop versions that have years of previous feature investment and already target multiple platforms.

7.2.2.2.2 RISING STAR DEVELOPER GALLERY APPS

These gallery apps don't have or necessarily require the larger development investments of top-tier apps, nor do they provide feature parity. These developers want to take advantage of as many built-in platform solutions as possible and will likely concentrate on a smaller amount of value-adds in the app. They are more likely to just use what is in the platform if it meets their basic needs. The monetization path for these apps is typically just selling the app itself. Because these apps are more dependent on system components for data, their ability to aggregate data across un-indexed data sources might be limited, and opportunities to optimize and customize the app for specific scenarios is likewise limited.

7.2.2.3 GUIDANCE FOR TOP TIER COMPETITIVE GALLERY APPS

- **Browsing**
    - The app should provide a custom database to ensure a consistent browsing experience that performs well.
    - This database needs to cache information (such as thumbnails, metadata, context, timestamps, and custom properties) from all the sources it cares about, such as the local PC, online storage, and so on, to provide a consistent user experience.
    - When consuming local content, the app should use the File Broker query APIs to import the important properties into the custom database.
    - If the app needs properties that aren't available from the File Broker query APIs, it should use the File broker API to access and extract properties for the database directly from the files.
- **Sorting and filtering**
    - To guarantee consistent performance, fast sorting, pivoting, and filtering should be provided by the app's custom database.
    - An alternate option is to use the File Broker query APIs to create a quick (but potentially inaccurate) view. This can help increase perceived performance, but at the risk of items being out of sync or unavailable if they are un-indexed.
- **Search**
    - Search should be driven by the local custom app database.
    - An alternate option is to use the File Broker query APIs, but the data might be out of sync, especially when the app aggregates data from non-indexed locations. This approach only works for supported file types.
- **Type Ahead for search**

        - The app should merge local and service-enabled type ahead (if the gallery covers local content) when providing type ahead.

7.2.2.4 GUIDANCE FOR RISING STAR DEVELOPER GALLERY APPS

- **Browsing**
    - The app can use the Windows system index via the File Broker to directly build a browsing view. (Note that this approach will be slower than querying a custom database.)
    - The File Broker supports the following query-backed views:
        - Generic: Folder, Date
        - Music: Song, Artist, Album, Title, Genre, Composer, Album Artist, Recorded Month
        - Photo: Date Taken, Months, Years, Tags, Ratings
- **Sorting and filtering**
    - Sorting and filtering can be accomplished by using the File Broker query-backed views (shapes) built on the system indexer. (Note: shapes are non-extensible lists of index-backed views defining how data can be returned to the app from the broker.)
- **Search**
    - Search can be provided by using the File Broker query APIs.
    - Search results can also be returned in any of the query backed-view (shapes) arrangements.
- **Typeahead for search**
    - If the app only displays local content, use local type ahead.
    - Otherwise, the type ahead provided should be a merge of local and service-enabled type ahead.

7.2.2.5 SUMMARY

App developers should carefully consider the trade-offs listed in the preceding sections before assuming one of the two solutions will be right for their users. Both solutions have trade-offs and limitations.

Using the File Broker\indexer as your app database and data collector:

Pros:

- Less work: leverages existing system components and services.
- Easy to use for quick simple apps that are not designed to be full-blown media library apps that compete with desktop apps.

Cons:

- There might be performance constraints.
- There are fewer customization possibilities.
- Not all data sources are indexed.

Building your own custom app database and data collector:

Pros:

- Provides more control over performance and customization.
- Is better for aggregated views from multiple data sources.

Cons:

- Requires more work: involves building and managing a database and keeping it in sync with the file system and other data sources.
- Requires more overall system resources to initially fill the database by harvesting data from the file system and other data sources.

7.3 WORKING WITH DEVICES

The Windows device ecosystem is open and diverse. Windows 8 helps modern application developers tap into this ecosystem by ensuring that the devices developers want to use are available on Windows systems, and it makes using such devices familiar and straightforward. Windows 8 also makes it easy for modern application developers to take advantage of innovative devices to create specialized, vertical solutions. Windows 8 provides a lightweight system that balances developer simplicity with the need to provide users with the confidence to let modern applications access their devices.

7.3.1 A TAXONOMY OF DEVICE FUNCTIONALITY

Not all device functionality is the same. This simple taxonomy helps illustrate how Windows 8 has approached factoring and delivering device functionality for modern application developers.

7.3.1.1 CORE DEVICE FUNCTIONALITY

Modern application developers just expect certain device functionality to be available on every Windows system. Such functionality includes mouse and keyboard input, graphics output, and audio input and output. Microsoft works together with its hardware partners to ensure that almost all systems available in the market support a core hardware profile. Windows itself relies on such device technology. Windows APIs are available to developers to ensure consistent and common behavior across partner hardware.

7.3.1.2 MODERN DEVICE FUNCTIONALITY

Modern application developers increasingly expect to rely on newer device functionality that has become commonplace, especially mobile form-factors. Such functionality includes location/GPS, webcam, multi-touch input, accelerometer, and near-field proximity. Microsoft works together with its hardware partners to ensure that appropriate systems in the market support a modern hardware profile. For such device functionality, Windows APIs are available to developers to ensure consistent and common behavior across partner hardware.

7.3.1.3 EVERYDAY DEVICE FUNCTIONALITY

Users often bring hardware to their system for use in their everyday life. Modern application developers might want to create tailored experiences for these everyday devices. Some device functionality, such as printing and removable storage and remote media renderers, is interesting to many classes of applications. Other device functionality, such as smart card and mobile broadband modems, is interesting only to specialized classes of applications. Microsoft works together with its hardware partners to ensure that their everyday hardware solutions work well with Windows. For such device functionality, Windows provides in-box APIs to ensure consistent and common behavior for partner hardware.

7.3.1.4 INNOVATIVE DEVICE FUNCTIONALITY

Users might also add hardware to their systems that isn't already supported by Windows. Modern application developers who target such hardware usually do so because they also sell the hardware, or because they have a business relationship with the provider of the hardware. Windows enables developers to access innovative hardware with ease when they explicitly target the hardware and develop for it. Microsoft works together with hardware partners who provide innovative solutions to ensure they work well with Windows.

7.3.2 PROGRAMMING MODEL CONSIDERATIONS

When designing how modern application developers access devices in Windows 8, several factors were taken into consideration.

7.3.2.1 TARGETING FRAMEWORKS

As described previously, modern applications can be developed in several different languages and presentations frameworks. And it's primary goal to make it easy and familiar for modern application developers to be able to work with devices. To meet that goal, Windows uses these guidelines in exposing device functionality to developers:

- If the functionality is **heavily tied into framework abstractions** (notably presentation and media) and needs to interact with existing framework data structures, the API into that framework. In these cases, there is usually an existing framework that provides the needed functionality. For Windows 8, these frameworks are adapted to support modern application development. One example is the Windows Printing API, which provided the foundation for XAML/C++ and HTML/JS/CSS versions of the Printing framework.

- If there is functionality in a framework due to **existing, well-established standards** considerations (this is common with HTML/JS/CSS), Windows implements he standard for that framework. Because there often isn't a well-established API for all frameworks, the API might be implemented in a broadly-adapted way within the Windows Runtime. An example is the Windows Location API, which contains an implementation of the W3C-defined GeoLocation API; another example is the Windows Webcam API, which adds webcams as a source for the HTML5 video tag.
- If the functionality has **no existing model** in any of the frameworks, then the Windows APIs are implemented directly within the Windows Runtime so that they are available to all supported frameworks. If the functionality has well-established abstractions in all existing frameworks and there is no need for a new API (because there are no shortcomings in these current abstractions), then the existing APIs are adapted for the Windows Runtime.

7.3.2.2 DEALING WITH DUPLICATION

When adapting Windows APIs to developer frameworks, duplication can be an issue. The following are some ways duplication issues have been dealt with:

- When an API does not make sense for a particular framework (for example, threading in JavaScript), it has been omitted from that framework. In this case, duplication is avoided.
- An existing Windows API may not have been appropriate to carry forward. So we would implement a new API in the Windows Runtime and adapt this to all frameworks. There would be no duplication in such a case and the old API would not be available for use via WinRT projections.
- There might be an existing API that must be supported (such as standards APIs) but a Windows API also exists that provides additional functionality. Duplication is allowed in such cases.

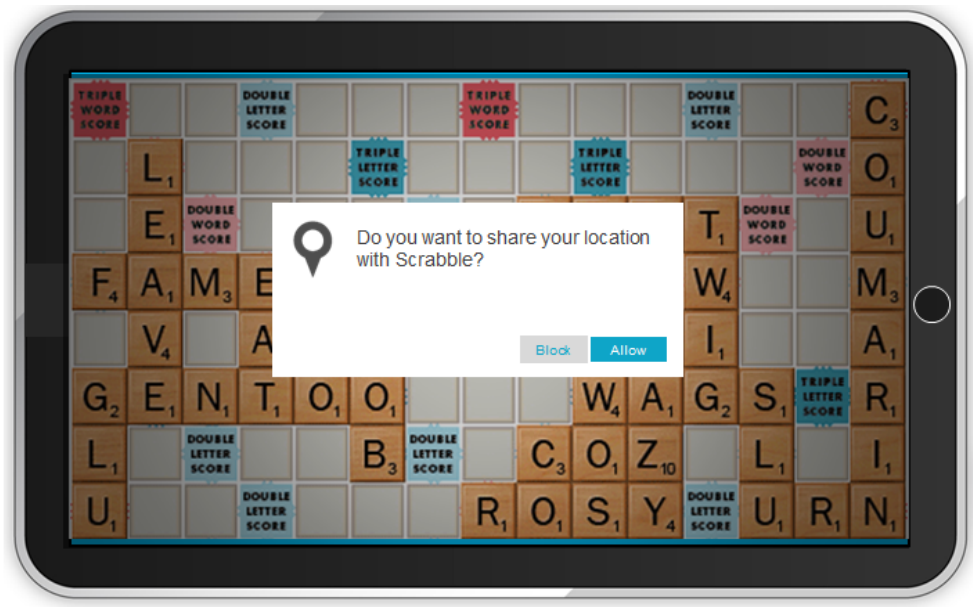
7.3.3 USER CONFIDENCE IN DEVICE ACCESS

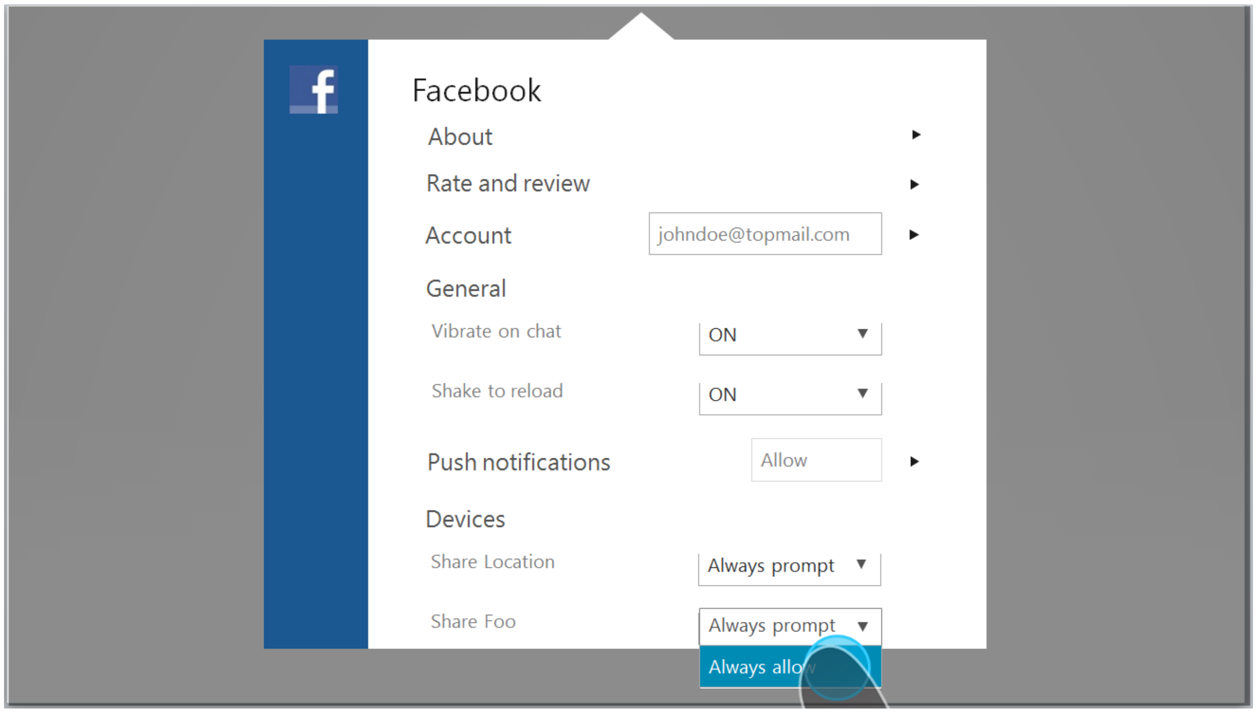
Devices form an integral part of a user's PC experience. Devices such as webcams, printers, and cameras are commonly used and are expected to work flawlessly with the PC. Yet, in a modern, connected environment, using some of these devices may concern users. Some devices, such as the webcam and microphone, disclose Personally Identifiable Information (PII). Other devices, such as the broadband modem and SMS, use metered services that may cost a significant amount of money.

The Windows 8 application model categorizes devices as base trust, sensitive, or custom.

- **Base trust devices** are those that users can be confident in Windows developers just being always allowed to use. These devices, such as printers and accelerometers, do not present a threat to the user in terms of disclosing PII or costing money.

- **Declared devices** are those that users can be confident in Windows developers being allowed to use, but only where the developer has declared the use of the device. This declaration is disclosed in the Windows UI to transparently help users know when an app needs to use a specific kind of device.
- **Sensitive devices** require that the application declare the capability to access them, and the user must provide consent to let the application use them. Consent is managed on a per-application, per-device-class basis. A consent UI that helps the user understand the specific device capability being requested is presented to the user at device access time. This lets the application present context in the application flow prior to the access attempt to help the user understand why the access should be granted (for example, consider a mapping application trying to access the PC's location). At any time, the user may use system UI to remove access to a sensitive device class from an application. It's important for developers to know what devices are sensitive and to design their applications to account for this revocation possibility.
- **Custom devices** are those devices that Windows doesn't know much about. Through a set of declarations associated with the device driver and in the application manifest, only the application that goes with the custom device is allowed to use it. Users can have confidence that no other application can use the device otherwise.


Do you want to share your location with Scrabble?


Facebook

About

Rate and review

Account johndoe@topmail.com

General

Vibrate on chat ON

Shake to reload ON

Push notifications Allow

Devices

Share Location Always prompt

Share Foo Always prompt

Always allow

7.3.4 LIST OF SUPPORTED APIS

The following list of device access APIs are currently supported for Windows 8. Where the declaration policy is either "Consent Required" or "Declaration Required", the device capability must be declared in the application manifest. Where the declaration policy is "Always Allowed", no declaration need be made.

Description	WinRT Namespace	Declaration Policy	Manifest Declaration
Microphone	Windows.Devices.Audio.In	Consent Required	**Microphone**
Webcam	Windows.Devices.Video.In	Consent Required	**Webcam**
Location	Windows.Devices.Geolocation	Consent Required	**Geolocation**
Phone text message	Windows.Devices.Wpd.Sms	Consent Required	PortableSms
PC text message	Windows.Devices.Sms	Consent Required	**PcSms**
Removable storage	Windows.Storage	Declaration Required	**RemovableStorage**
Near field proximity	Windows.Networking.Proximity	Declaration Required	**NearFieldProximity**
Portable	Windows.Devices.Wpd	Declaration Required	GUID

device			
Portable device contacts	Windows.Devices.Wpd.Contacts	Declaration Required	GUID
Portable device calendars	Windows.Devices.Wpd.Calendar	Declaration Required	GUID
Portable device tasks	Windows.Devices.Wpd.Tasks	Declaration Required	GUID
Portable device notes	Windows.Devices.Wpd.Notes	Declaration Required	GUID
Portable device status	Windows.Devices.Wpd.Status	Declaration Required	GUID
Portable device ringtones	Windows.Devices.Wpd.Ringtones	Declaration Required	GUID
Accelerometer	Windows.Devices.Sensors.Accelerometer	Always Allowed	**n/a**
Inclinometer	Windows.Devices.Sensors.Inclinometer	Always Allowed	**n/a**
Compass	Windows.Devices.Sensors.Compass	Always Allowed	**n/a**
Gyrometer	Windows.Devices.Sensors.Gyrometer	Always Allowed	**n/a**
Orientation sensor	Windows.Devices.Sensors.OrientationSensor	Always Allowed	**n/a**
Simple orientation sensor	Windows.Devices.Sensors.SimpleOrientationSensor	Always Allowed	**n/a**
Light sensor	Windows.Devices.Sensors.LightSensor	Always Allowed	**n/a**
Printer	Windows.Devices.Print	Always Allowed	**n/a**

7.3.5 VERTICAL SCENARIOS: DEVICE INNOVATION AND DEVICE COMPANION APPLICATIONS

Modern application developers may want to build a vertical solution that combines their application with innovative hardware and services on the Internet. Windows 8 supports two technologies that make such products easier to create and distribute.

7.3.5.1 CUSTOM DEVICE ACCESS

The Application Container model denies system resources to the modern developer by default. As discussed earlier, device stacks built in to Windows broker device access to modern applications depending on the device capabilities model and, if necessary, user consent.

Windows supports device developers obtaining brokered access through the device I/O manager to a custom driver implementing a proprietary device interface class declared as a trusted capability in the application manifest. It is the responsibility of the application developer to write to the device interface. The best-practice is to create a MoCOM extension and deliver it with the modern application. This approach makes it easy to develop an application that uses the custom device with any framework.

Custom device access is supported only by the privileged application designed to work with the device. No other application may access the device. Privileging of the specific device is done through device metadata associated with the custom device driver.

7.3.5.2 DEVICE COMPANION APPLICATION

A Device Companion Application (DCA) is a modern application that is tightly associated with a particular device or family of devices. Windows can support automatically and seamlessly installing a DCA at the time a device is installed with Windows. The DCA can work with in-box device capabilities or custom device capabilities.

When a user connects a device to Windows or discovers the device on the network and initiates pairing with the device, Windows determines the device drivers to install for the device. If the user has consented to let Windows look online to complete device setup, Windows obtains any necessary drivers and installs them. Then, if the device maker has created a DCA for the device and put this into the Store, Windows will also find and install that application. Once the application is installed, the user is invited to launch it.

A DCA can be developed just like any other modern application. The application must meet certain requirements that are validated at application onboarding to ensure the application's close association with the device (or family of devices). The device and its associated device metadata must also pass WHQL requirements.

7.3.6 EXAMPLES OF WORKING WITH DEVICES

7.3.6.1 USING LOCATION

Look here for the latest on using the Navigator.Geolocation and Windows.Devices.Geolocation namespaces:

http://windows/windows8/dnt/dcon/DCON%20Team%20WIKI/Location.aspx

7.3.6.2 USING SENSORS

Look here for the latest on using the Sensor APIs in Windows:

http://windows/windows8/dnt/dcon/DCON%20Team%20WIKI/Windows%20Runtime%20Sensors.aspx

7.3.6.2.1 TYPICAL USE SCENARIOS FOR SENSORS

Accelerometer

- Handheld device gaming (PC as handheld device)
- Shake gestures

Inclinometer

- Handheld device gaming (PC as handheld device)
- Handheld device as tool (a level)
- Device orientation detection

Compass

- Navigation using handheld device (PC as handheld device)

Light Sensor

- Modify application display characteristics based on lighting conditions (e.g., night view)
- Automatic color correction

7.3.6.2.2 SENSOR APIS

Windows.Devices.Sensors.Accelerometer

Windows.Devices.Sensors.Inclinometer

Windows.Devices.Sensors.Compass

Windows.Devices.Sensors.LightSensor

7.3.6.2.3 JAVASCRIPT EXAMPLE (ACCELEROMETER)

```
<html>
<head>
    <title>WinRT Inclinometer Sample</title>
    <script type="text/javascript" language="javascript">
        function outputReport(e) {
           /* Example output for an accelerometer at rest at sea level:
            *  {"Timestamp":712534723,"AccelerationX":0,"AccelerationY":0,
"AccelerationZ":1}
            */

            document.getElementById("readingOutput").innerHTML =
JSON.stringify(e);
        }

        function onDataChanged(e) {
            outputReport(e);
        }

        document.addEventListener("DOMContentLoaded", onReady, false);
        function onReady() {
            var sensor =
Windows.Devices.Sensors.Accelerometer.getDefault();
            if ( sensor != null ) {
                sensor.addEventListener("readingchanged", onDataChanged,
false);
            }
        }
    </script>
</head>
<body>
<a href="#" id="readingOutput" style=""></a>
</body>
</html>
```

7.3.6.2.4 JAVASCRIPT EXAMPLE (INCLINOMETER)

```
<html>
<head>
    <title>WinRT Inclinometer Sample</title>
    <script type="text/javascript" language="javascript">
        function outputReport(e) {
           /* Example output for an inclinometer on a horizontal surface:
            *  {"Timestamp":712534723,"TiltInDegreesX":0,"
TiltInDegreesY":0," TiltInDegreesZ":0}
            */
            document.getElementById("readingOutput").innerHTML =
JSON.stringify(e);
        }

        function onDataChanged(e) {
            outputReport(e);
        }

        document.addEventListener("DOMContentLoaded", onReady, false);
        function onReady() {
            var sensor =
Windows.Devices.Sensors.Inclinometer.getDefault();
            if ( sensor != null ) {
                sensor.addEventListener("readingchanged", onDataChanged,
false);
                outputReport(sensor.getCurrentReading());
            }
        }
    </script>
</head>
<body>
<a href="#" id="readingOutput" style=""></a>
</body>
</html>
```

7.3.6.2.5 JAVASCRIPT EXAMPLE (COMPASS)

```
<html>
<head>
    <title>WinRT Compass Sensor Sample</title>
    <script type="text/javascript" language="javascript">
        function outputReport(e) {
           /* Example output for a compass on a horizontal surface,
heading magnetic north,
            *  with declination of zero, and with tilt compensation:
            *  Heading (TrueNorth) = 0,
            *  TiltCompensated
            */
            var t = "Heading (MagneticNorth) = " + e.headingMagneticNorth;
            var caps = e.details.capabilities;

            document.getElementById("readingOutput").innerHTML = t;
        }

        function onDataChanged(e) {
            outputReport(e);
        }

        document.addEventListener("DOMContentLoaded", onReady, false);
        function onReady() {
            var sensor = Windows.Devices.Sensors.Compass.getDefault();
            if ( sensor != null ) {
                sensor.addEventListener("readingchanged", onDataChanged, 
false);
                outputReport(sensor.getCurrentReading());
            }
        }
    </script>
</head>
<body>
<a href="#" id="readingOutput" style=""></a>
</body>
</html>
```

7.3.6.2.6 JAVASCRIPT EXAMPLE (LIGHT SENSOR)

```
<html>
<head>
    <title>WinRT Light Sensor Sample</title>
    <script type="text/javascript" language="javascript">
        function outputReport(e) {
            /* Example output for a simple ambient light sensor in 
complete darkness:
            *  Illuminance = 0
            */
            var t = "Illuminance = " + e.illuminance;

            document.getElementById("readingOutput").innerHTML = t;
        }

        function onDataChanged(e) {
            outputReport(e);
        }

        document.addEventListener("DOMContentLoaded", onReady, false);
        function onReady() {
            var sensor = Windows.Devices.Sensors.LightSensor.getDefault();
            if ( sensor != null ) {
                sensor.addEventListener("readingchanged", onDataChanged, 
false);
                outputReport(sensor.getCurrentReading());
            }
        }
    </script>
</head>
<body>
<a href="#" id="readingOutput" style=""></a>
</body>
</html>
```

8 SAMPLE APPS AND ADDITIONAL LINKS

Sample apps – http://modernappstesting (see the Modern App Library link)

Wikis / FAQs – http://devxwiki, http://windows/windows8/uex/Partners/default.aspx

Alias for support / questions RE: building Windows 8 applications – dog8appbld

Writing high performance WWAs - http://windows/windows8/DevX/DevXWiki/Wiki%20Pages/WWA%20Performance.aspx

Performance in relation to WinRT UI Apps – http://windows/windows8/docs/Windows%208%20Feature%20Documents/Developer%20Experience%20(DEVX)/DevDiv%20Specs/Org/Jupiter/Fundamentals/Performance/

A list of the Windows RunTime APIs available to MoSh Application Developers - http://windows/windows8/DevX/rex/Shared%20Documents/Partner%20Engagement/API/Working%20-%20WinRT.xlsx