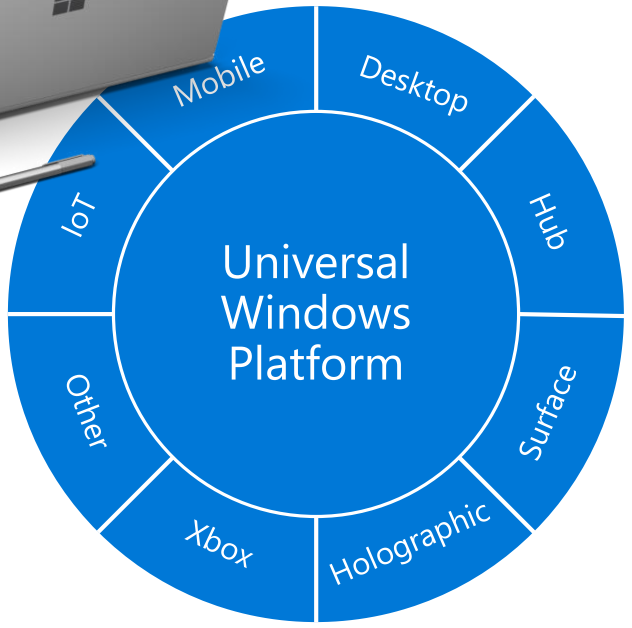
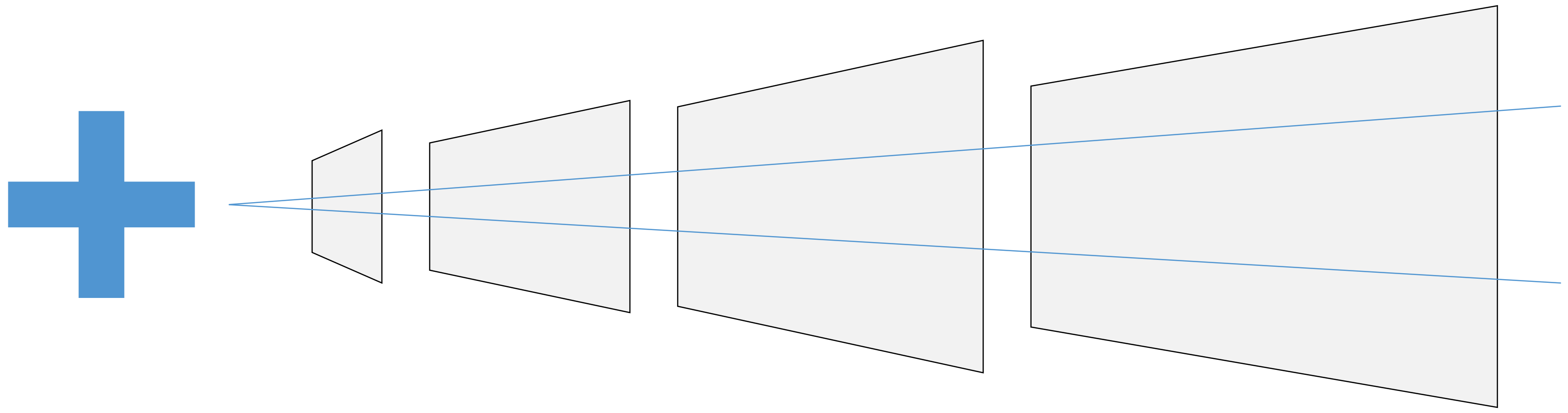
The Universal Windows Platform

App structure, page layout, and navigation are the foundation of an app's user experience. The Universal Windows Platform (UWP) makes is extremely easy to create a user interface that harmonizes a variety of devices with different display sizes, as well as foundational navigation paradigms that make sense across devices.



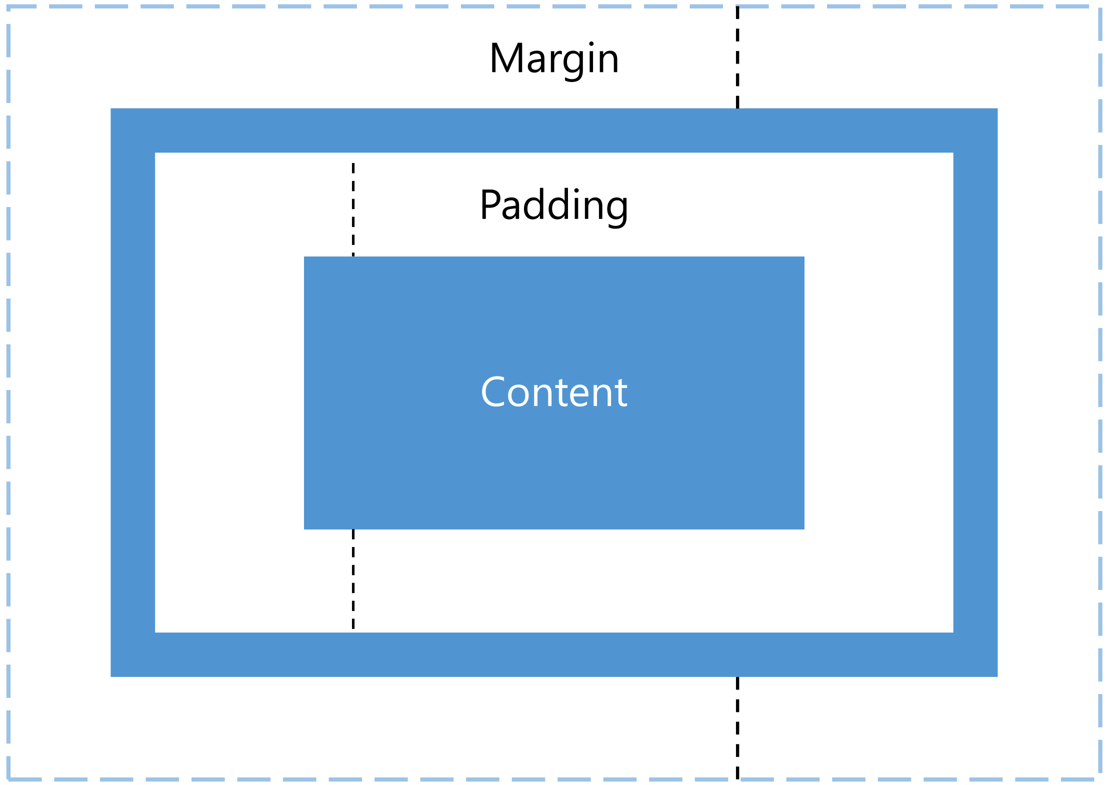
Adaptive UI Concepts

An app should have a highly adaptive UI and comprehensive input capabilities to enable it to function properly on wide variety of devices with an almost infinite number of form factors and orientations.



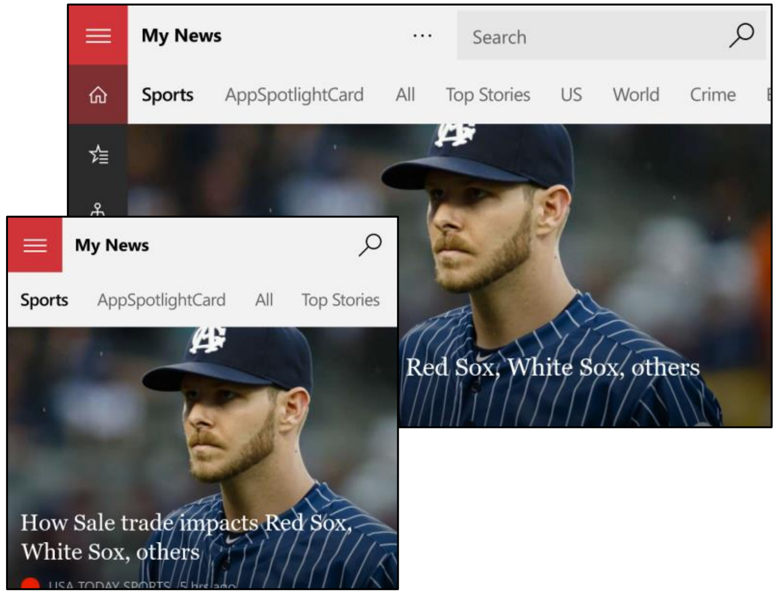
Layout Controls

The key to a responsive or "fluid" layout is the appropriate use of layout properties and panels to automatically and intelligently reposition, resize, and "reflow" content. With UWP you can set a fixed size on an element, or use automatic sizing to allow a parent layout panel handle sizing and flow.



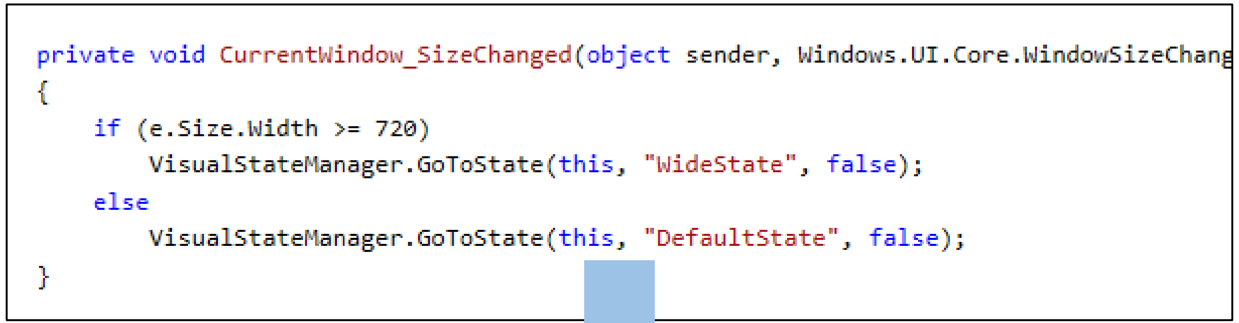
Adaptive Layouts

When an app window grows or shrinks beyond a certain amount, you might want to change layout properties to reposition, resize, reflow, reveal, or even totally replace sections of your UI. You can define different visual states for your UI, and then apply them when the window width or height crosses a specified threshold. An AdaptiveTrigger provides an easy way to set the threshold (or 'breakpoint') where a change in state is applied or becomes active.

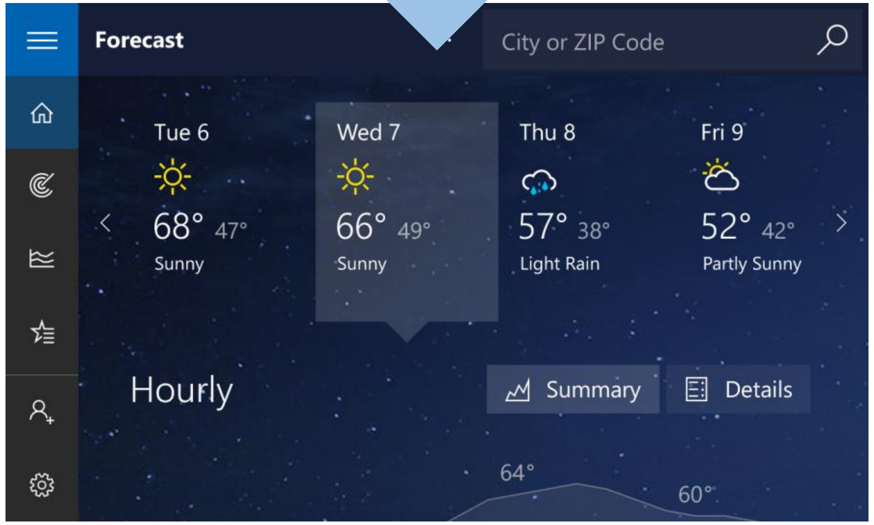


Visual States

Visual States represent the visual appearance of a UI element when it is in a specific state. Visual States use the concept of Setters or a Storyboard to set UI properties within user interface elements, and are nested in a Visual State Group for ease of manipulation and coordination.

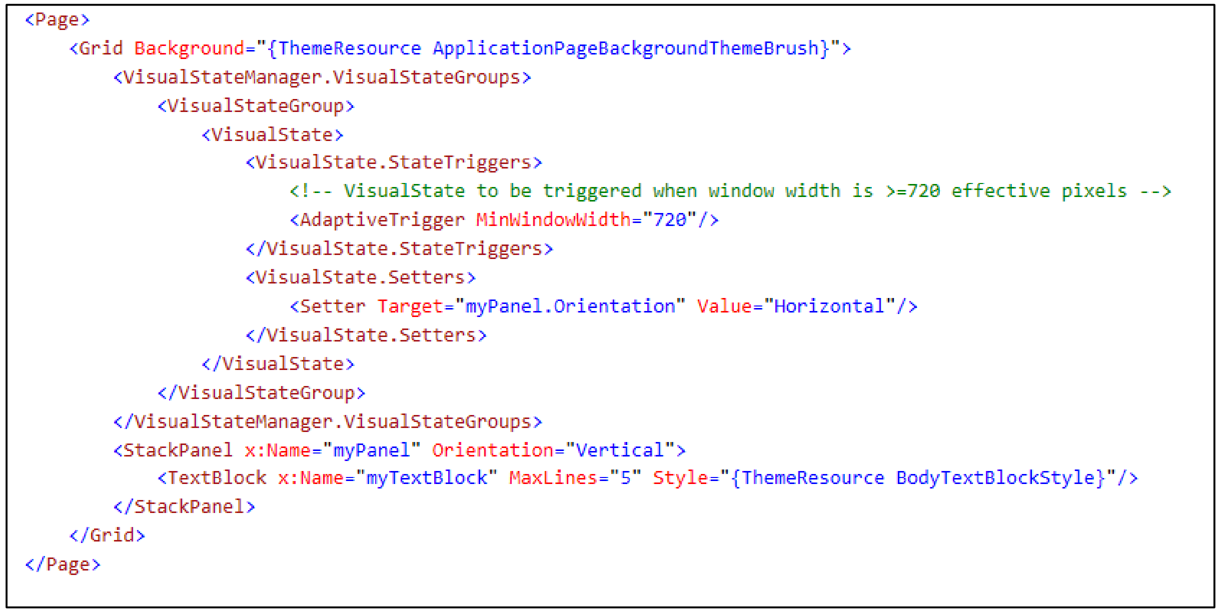


```
private void CurrentWindow_SizeChanged(object sender, Windows.UI.Core.WindowSizeChang
{
    if (e.Size.Width >= 720)
        VisualStateManager.GoToState(this, "WideState", false);
    else
        VisualStateManager.GoToState(this, "DefaultState", false);
}
```



Adaptive Triggers

An Adaptive Trigger creates a rule that automatically triggers a Visual State change when the window is a specified height or width. When you use Adaptive Triggers in your XAML markup, you no longer need to handle these changes programmatically.



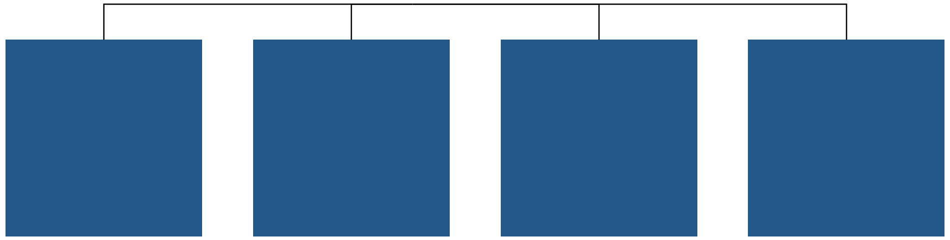
```
<Page>
    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <VisualStateManager.VisualStateGroups>
            <VisualStateGroup>
                <VisualState>
                    <VisualState.StateTriggers>
                        <!-- VisualState to be triggered when window width is >=720 effective pixels -->
                        <AdaptiveTrigger MinWindowWidth="720"/>
                    </VisualState.StateTriggers>
                    <VisualState.Setters>
                        <Setter Target="myPanel.Orientation" Value="Horizontal"/>
                    </VisualState.Setters>
                </VisualState>
            </VisualStateGroup>
        </VisualStateManager.VisualStateGroups>
        <StackPanel x:Name="myPanel" Orientation="Vertical">
            <TextBlock x:Name="myTextBlock" MaxLines="5" Style="{ThemeResource BodyTextBlockStyle}"/>
        </StackPanel>
    </Grid>
</Page>
```

Navigation Basics

Navigation in Universal Windows Platform (UWP) apps is based on a flexible model of navigation structures, navigation elements, and system-level features. Together, they enable a variety of intuitive user experiences for moving between apps, pages, and content.

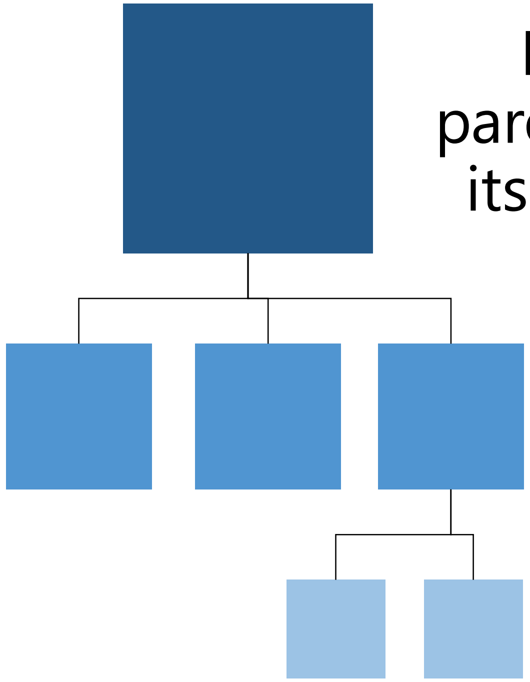
Navigation Paradigms

As peers



Between pages in the same level of the same subtree.

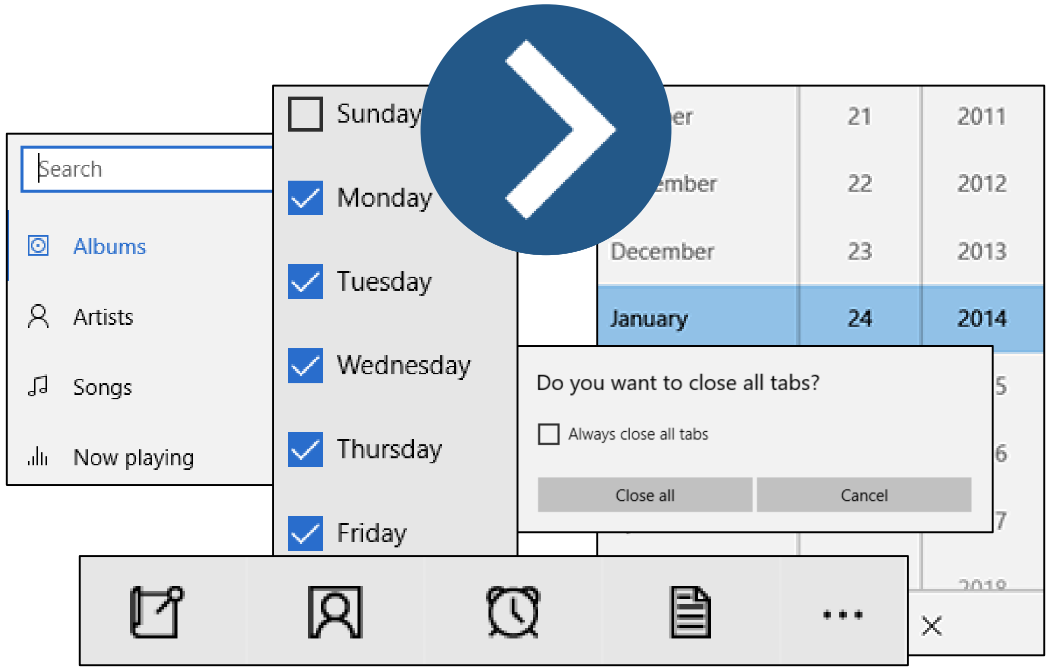
In a hierarchy



Between a parent page and its child pages

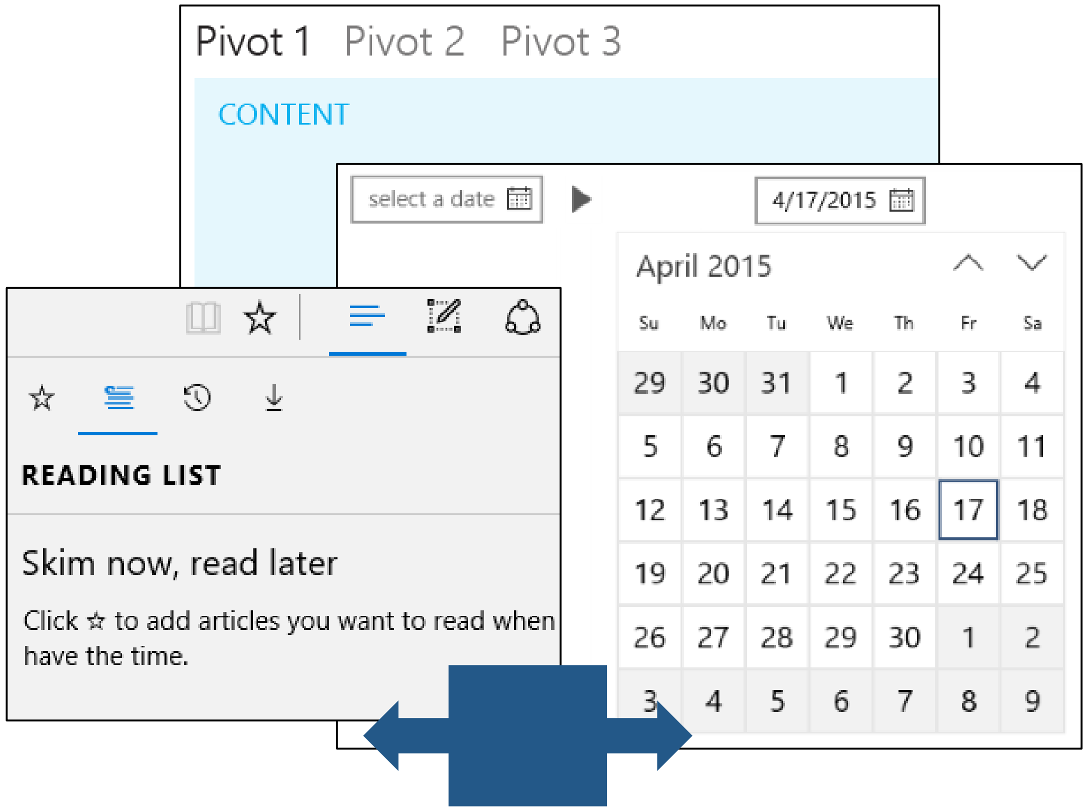
Common Controls

A control is a UI element that displays content or enables interaction. You create the UI for your app by using controls such as buttons, text boxes, and combo boxes to display data and get user input.



Control Patterns

A pattern is a recipe for combining several controls to make something new, such as combining a pivot control and labels to create a tab control, or a picker and a calendar control to create a date picker control.

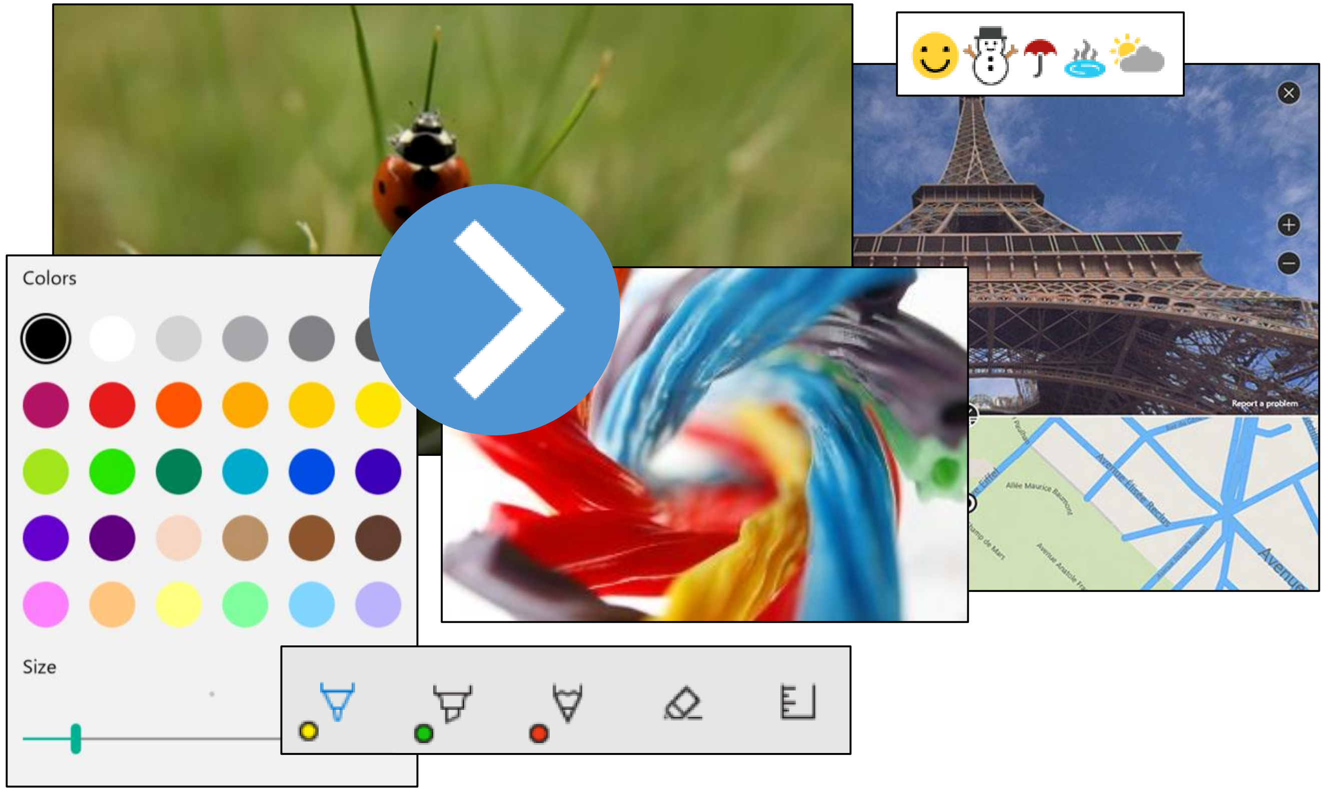


Skim now, read later

Click ☆ to add articles you want to read when have the time.

Advanced Controls

Advanced control support is defined as controls that target features and experiences beyond a standard text input or response, such as rich media, mapping, inking, speech recognition, or drill-down interactivity.

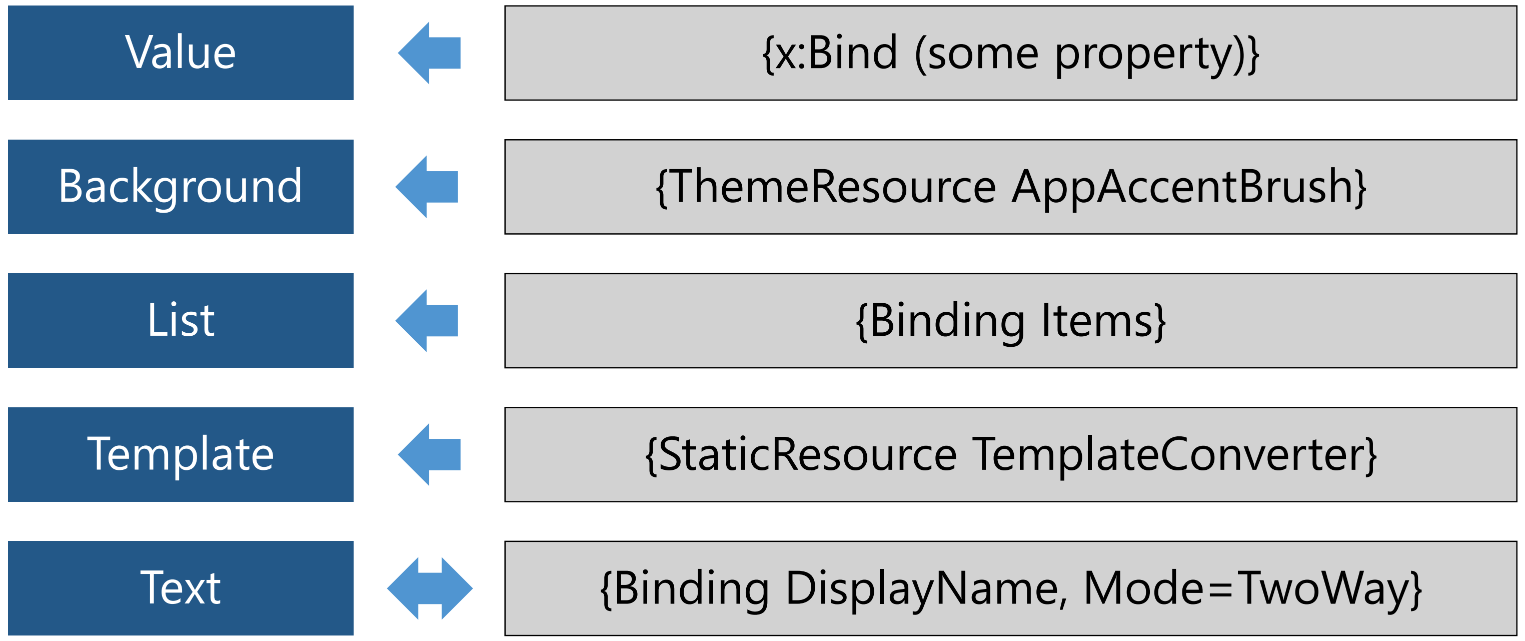


Binding Concepts

- Control labels to model values
- List collections to model lists
- Control properties to static values
- Setting controls to dynamic values
- Page values to local values
- Font families and brushes to resources
- Control actions to commands

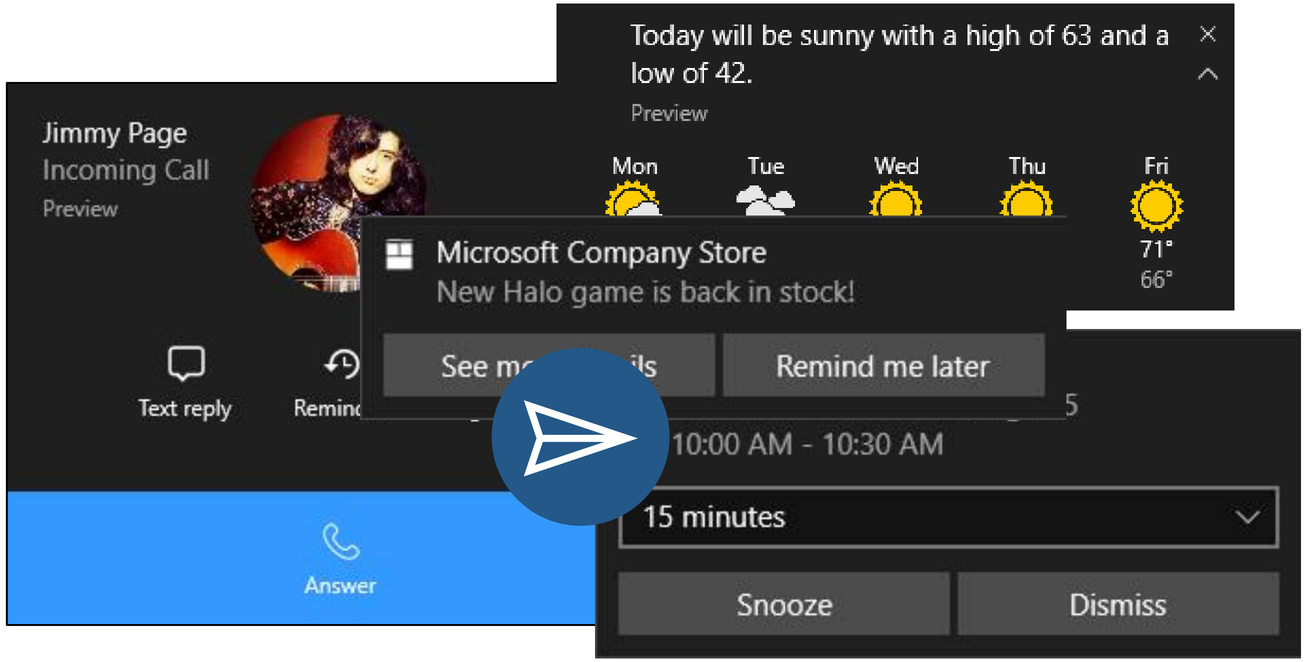
Binding Techniques

The term **binding** in this series (and in the Universal Windows Platform) does not always refer to “data” binding, per se, as it could refer to binding values to things like resources or other ancillary objects.


Value
{x:Bind (some property)}
Background
{ThemeResource AppAccentBrush}
List
{Binding Items}
Template
{StaticResource TemplateConverter}
Text
{Binding DisplayName, Mode=TwoWay}

Adaptive Toasts

Adaptive and interactive toast notifications let you create flexible pop-up notifications with more content, optional inline images, and optional user interaction, including the ability to add buttons, inputs, and scenario-based templates such as alarms, reminders, and incoming calls.



Today will be sunny with a high of 63 and a low of 42.

Preview

Jimmy Page

Incoming Call

Preview

Microsoft Company Store

New Halo game is back in stock!

10:00 AM - 10:30 AM

15 minutes

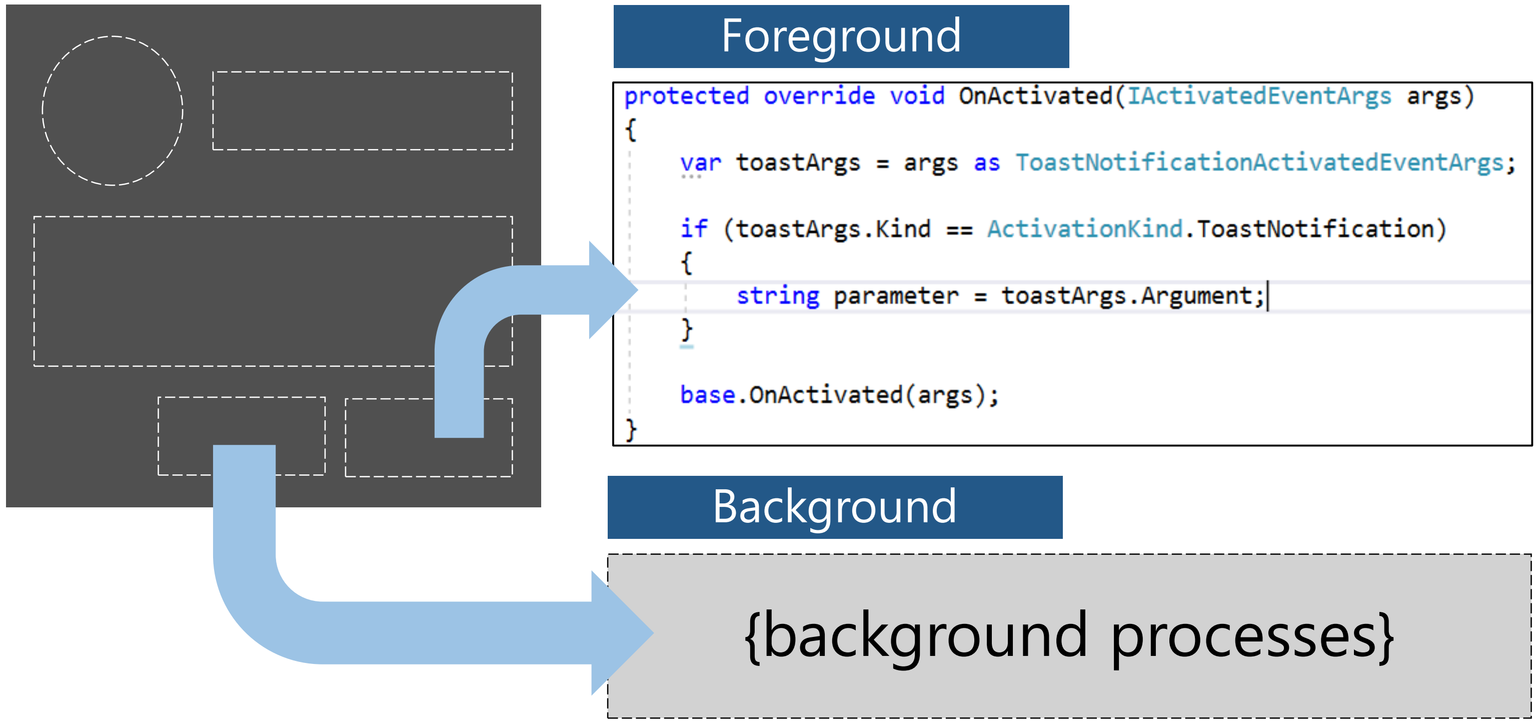
Delivery methods

Recommended delivery methods will change or even be combined with other methods based on the specific scenarios in your app.

Delivery method	Use with	Description
Local	Tile, Badge, Toast	A set of API calls that send notifications while your app is running, directly updating the tile or badge, or sending a toast notification.
Scheduled	Tile, Toast	A set of API calls that schedule a notification in advance, to update at the time you specify.
Periodic	Tile, Badge	Notifications that update tiles and badges regularly at a fixed time interval by polling a cloud service for new content.
Push	Tile, Badge, Toast, Raw	Notifications sent from a cloud server, even if your app isn't running.

Toast Interactivity

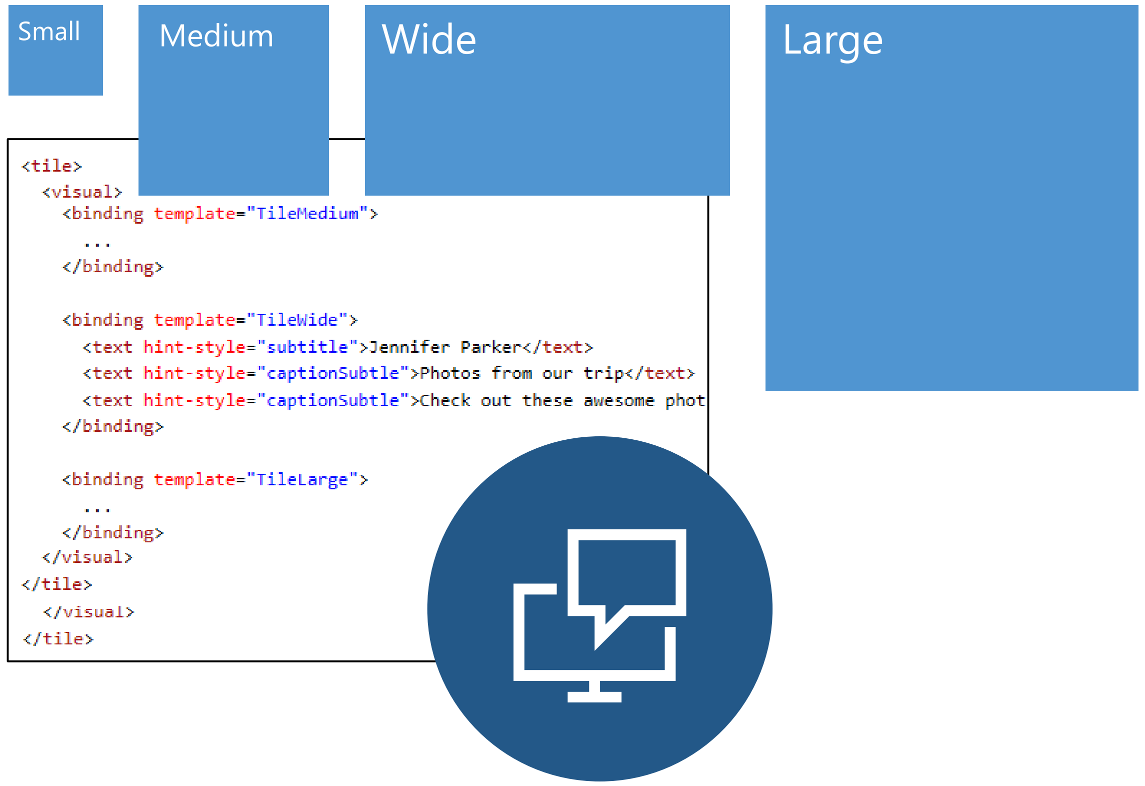
Interactive toast notifications are a new feature in the Universal Windows Platform that enable users to create toast notifications elements that enable user interaction on nested controls within a toast to respond to specific, parametrized events.



```
protected override void OnActivated(IActivatedEventArgs args)
{
    var toastArgs = args as ToastNotificationActivatedEventArgs;

    if (toastArgs.Kind == ActivationKind.ToastNotification)
    {
        string parameter = toastArgs.Argument;
    }

    base.OnActivated(args);
}
```

Adaptive Tiles

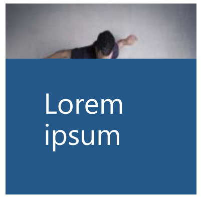
Adaptive tiles and tile templates allow you to design your own tile notification content using a simple and flexible markup language that adapts to different screen densities. Adaptive templates are designed to work across different form factors and notification types to create appropriate notifications based on device type.



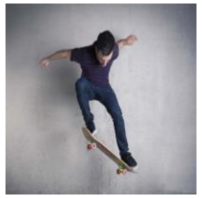
```
<tile>
  <visual>
    <binding template="TileMedium">
      ...
    </binding>

    <binding template="TileWide">
      <text hint-style="subtitle">Jennifer Parker</text>
      <text hint-style="captionSubtle">Photos from our trip</text>
      <text hint-style="captionSubtle">Check out these awesome phot
    </binding>

    <binding template="TileLarge">
      ...
    </binding>
  </visual>
</tile>
  </visual>
</tile>
```

Adaptive Tiles

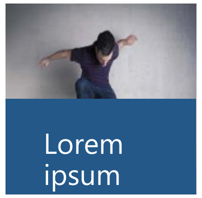
For a more engaging experience, you can specify an image that "peeks" in from the top of the tile. The peek image uses an animation to slide down/up from the top of the tile, peeking into view, and then later sliding back out to reveal the main content on the tile.



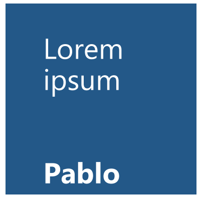
Animated



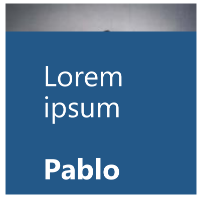
Peek shown



Peek sliding up



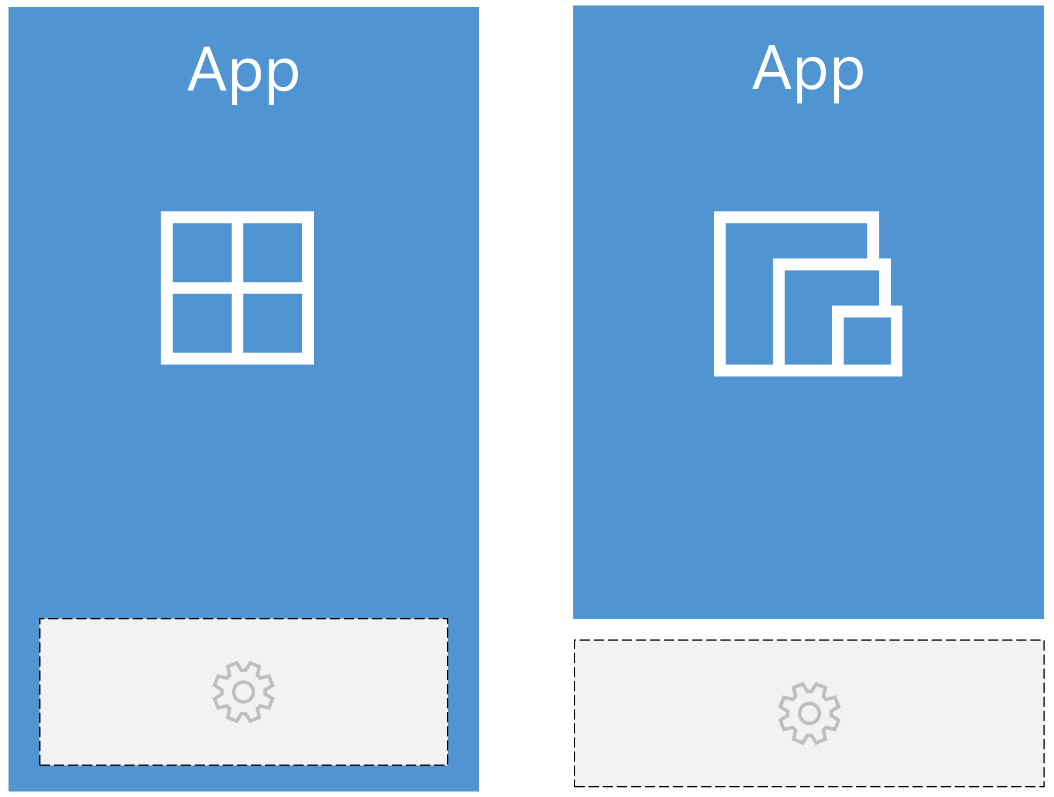
Content shown



Peek sliding down

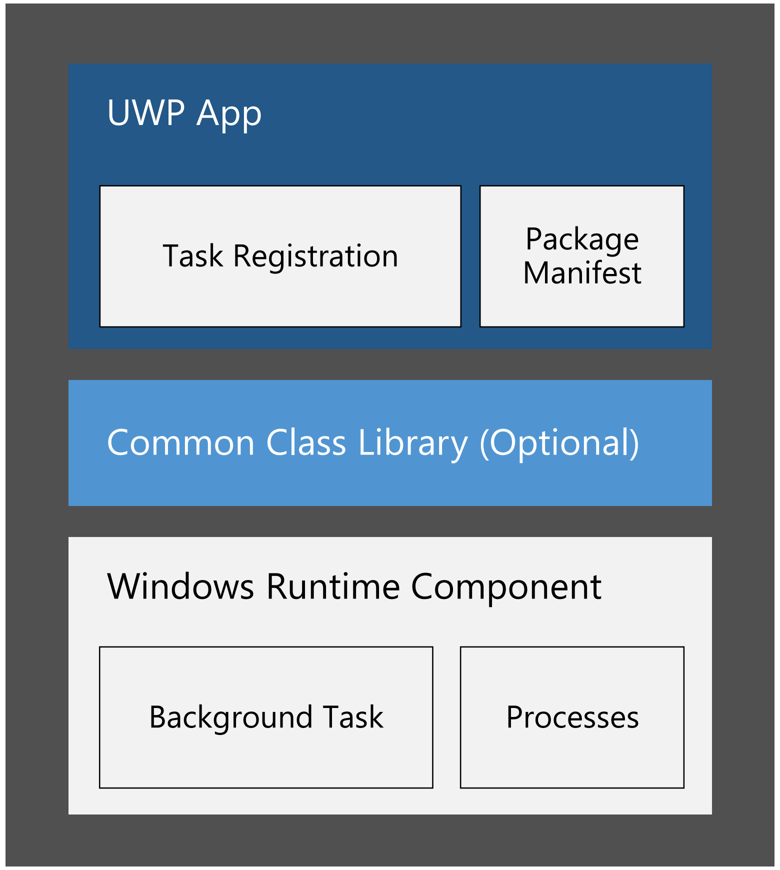
Background Task Concepts

There are two approaches to implementing background tasks: in-process, in which the app and its background process run in the same process; and out-of-process, where the app and the background process run in separate processes. In general, both approaches provide the same level of functionality, however out-of-process tasks are far more resilient.



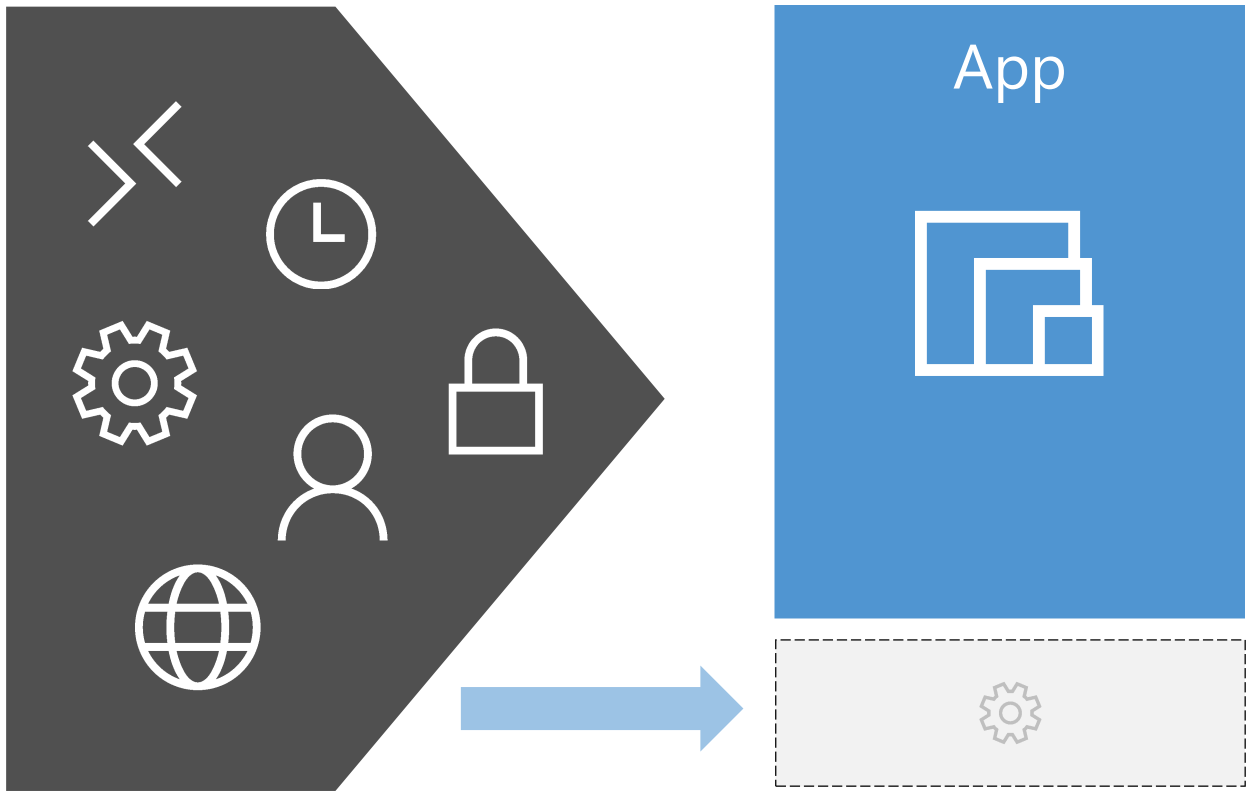
Background Task Dependencies

- Create a Windows Runtime Component
- Add a sealed class that implements IBackgroundTask
- Add a project reference to the Windows Runtime Component
- Add a package manifest entry to allow and direct communication
- Programmatically register the task
- Create programmatic processes to handle task instances and cleanup



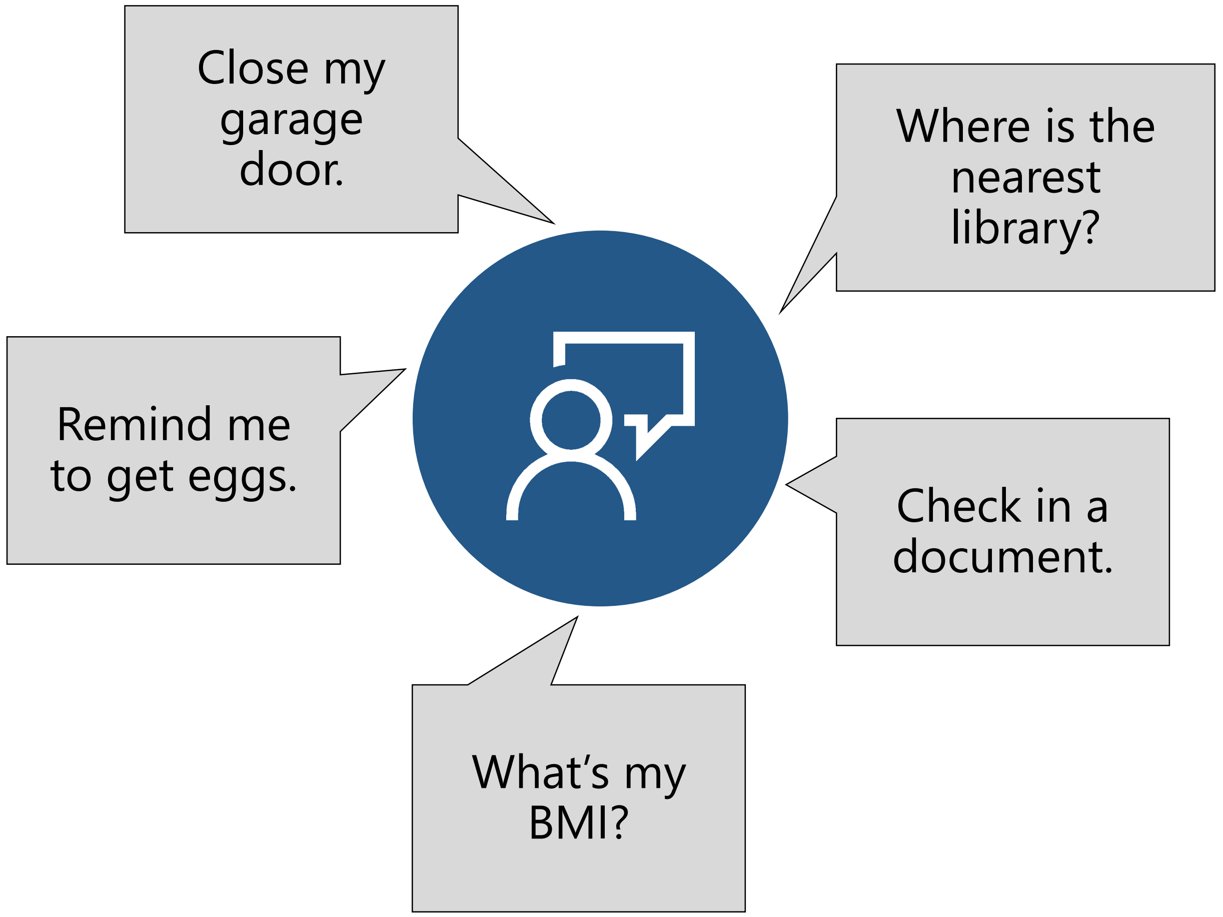
System Triggers

An app can respond to system-generated events by registering a background task with the System Trigger class. System Trigger tasks are perfect for apps with requirements around dates and times, as well as connectivity states and battery or power level fluctuations.



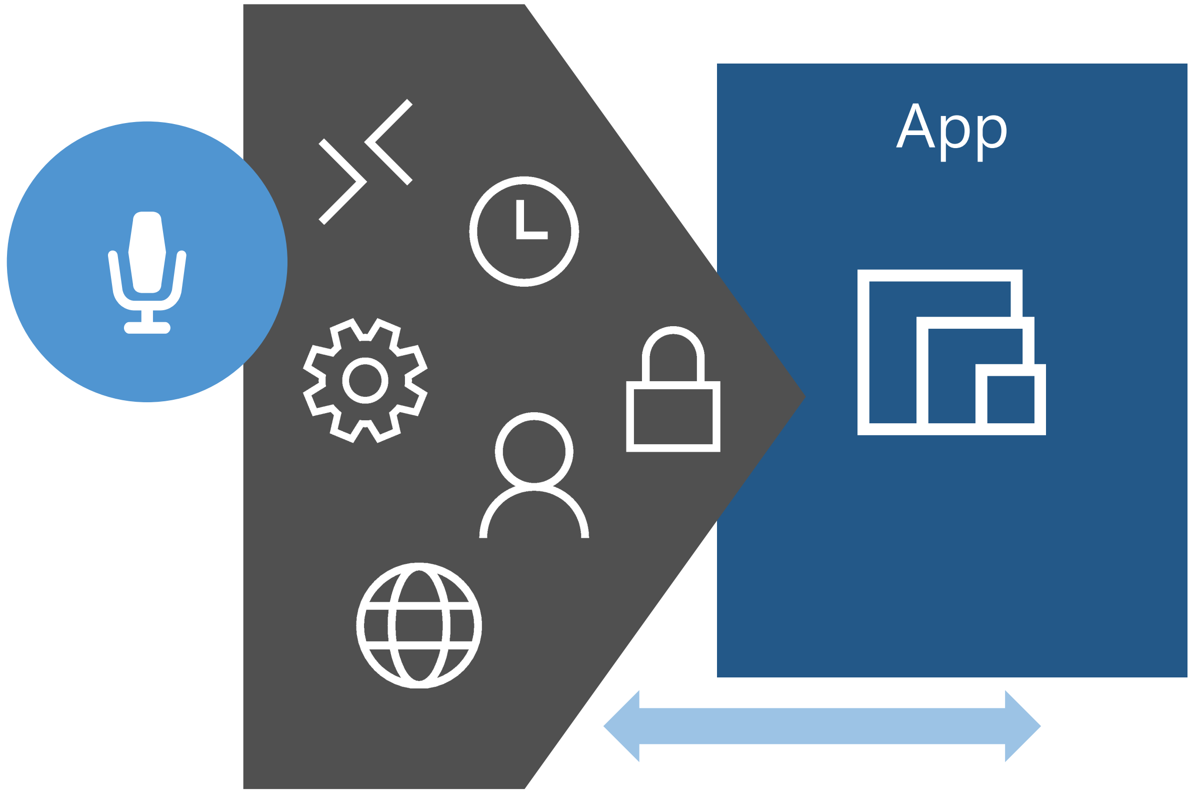
Voice and Speech Concepts

Simply put, the driving principle behind voice and speech in the Universal Windows Platform is to provide a robust, extensible framework where **any task** can be performed against **any source**, through speech recognition, natural language interpretation, and speech synthesis.



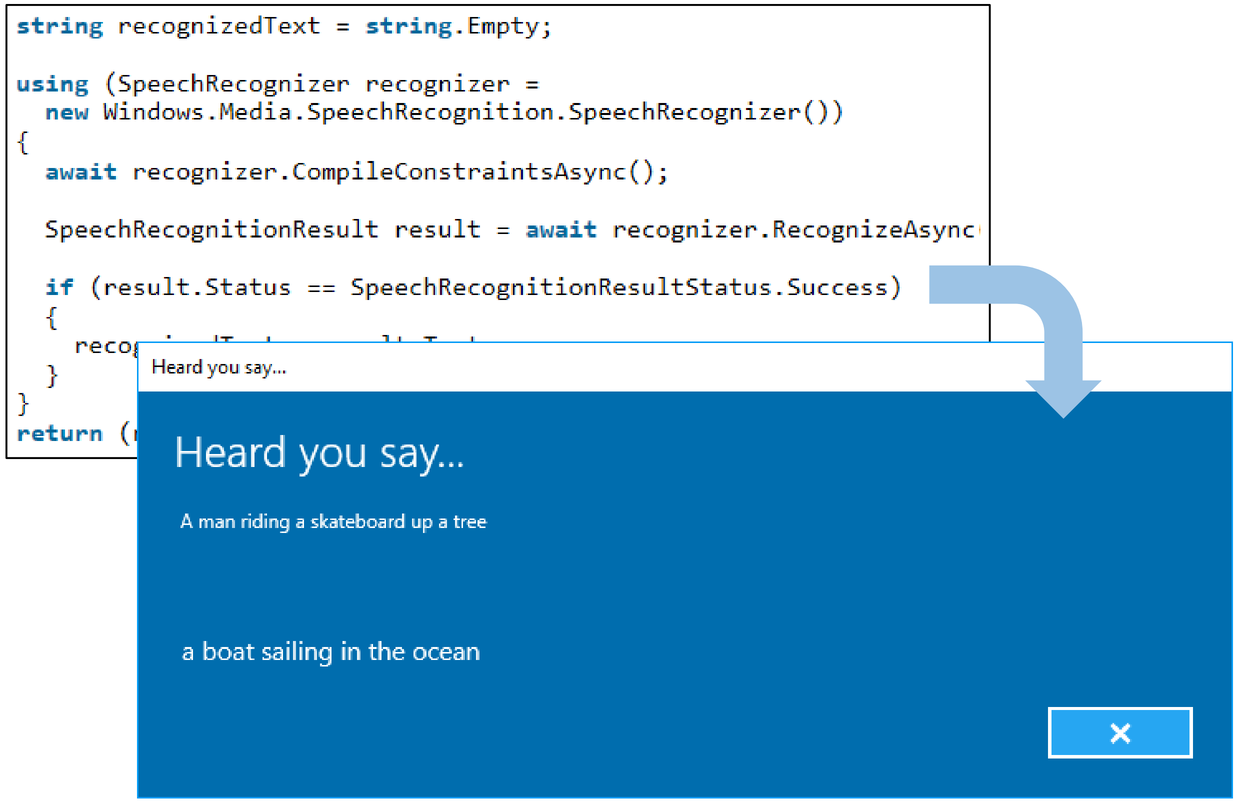
Common Scenarios

- Get information
- Initiate a process
- Search documents
- Add reminders
- Perform actions
- Guided assistance



The Speech Recognizer

Speech recognition is made up of a speech runtime, recognition APIs for programming the runtime, ready-to-use grammars for dictation and web search, and a default system UI that helps users discover and use speech recognition features, including text-to-speech (TTS), natural language understanding, commanding, as well as pre-defined and custom grammars.



```
string recognizedText = string.Empty;

using (SpeechRecognizer recognizer =
  new Windows.Media.SpeechRecognition.SpeechRecognizer())
{
  await recognizer.CompileConstraintsAsync();

  SpeechRecognitionResult result = await recognizer.RecognizeAsync

  if (result.Status == SpeechRecognitionResultStatus.Success)
  {
    recog
  }
}
return (
```

Heard you say...

Heard you say...

A man riding a skateboard up a tree

a boat sailing in the ocean

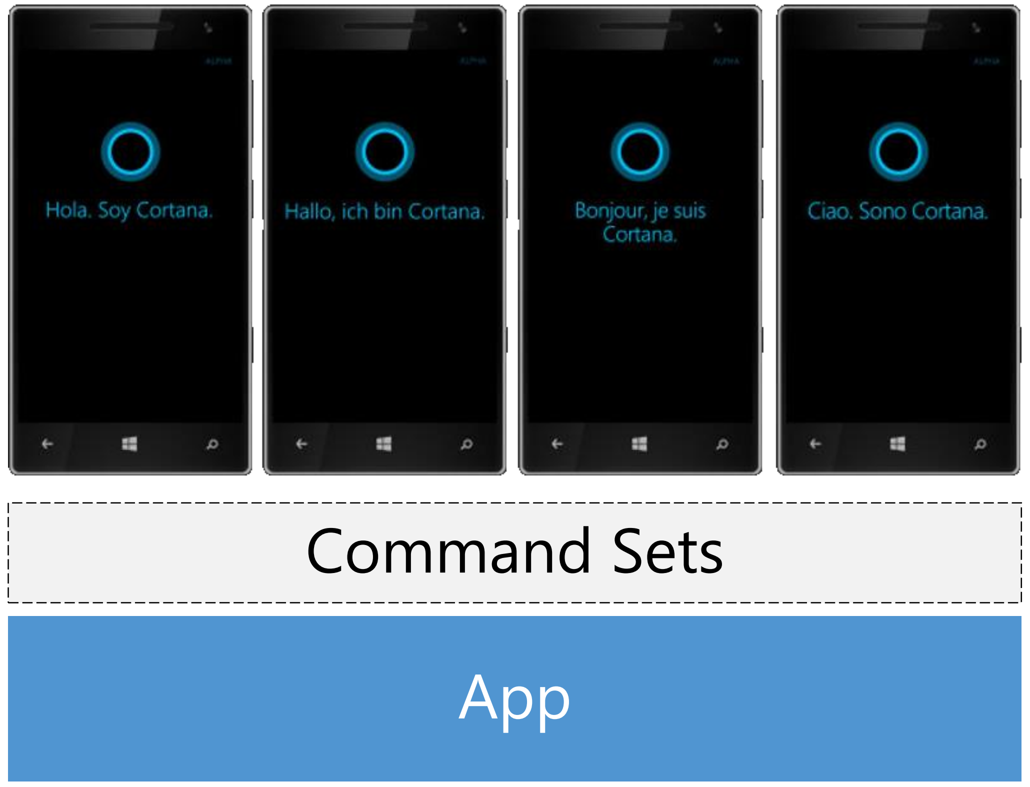
Speech Synthesis

The Universal Windows Platform provides support for initializing and configuring a speech synthesis engines to convert a text strings to audio streams, also known as text-to-speech (TTS). Voice characteristics (like gender), pronunciation, volume, pitch, rate or speed, and emphasis can also be controlled via Speech Synthesis Markup Language (SSML).

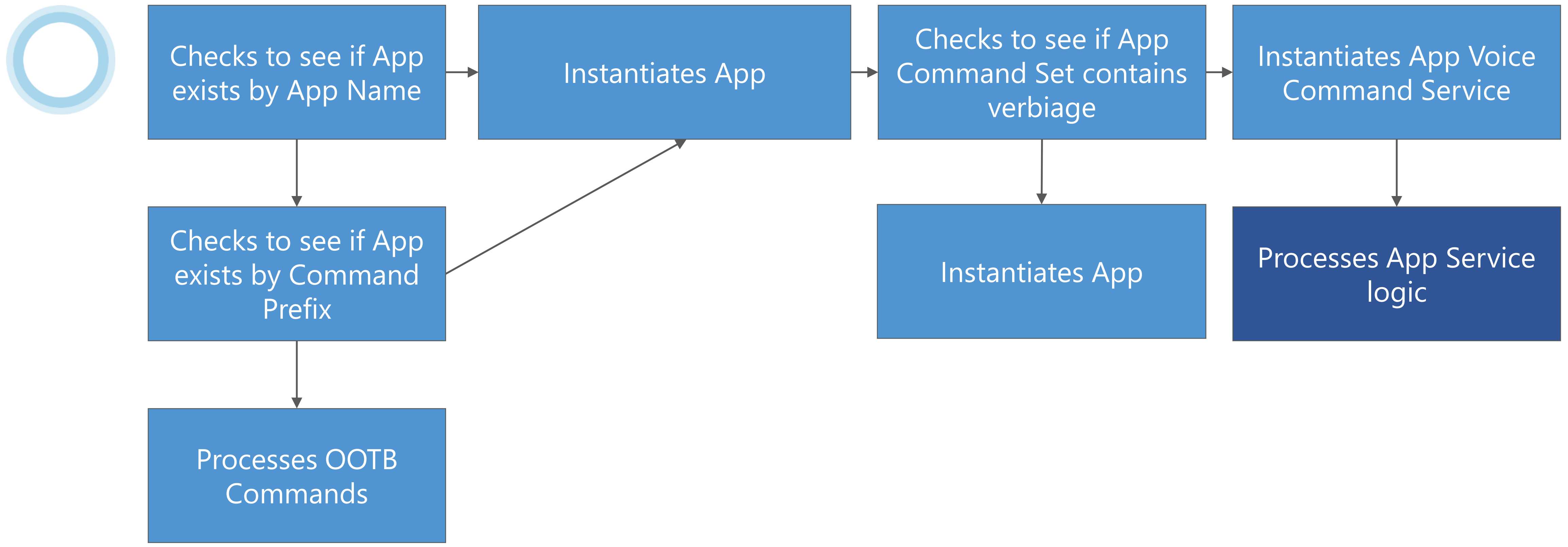
```
// The media object for controlling and playing audio.
MediaElement mediaElement = this.media;

// The object for controlling the speech synthesis engine (voice).
var synth = new Windows.Media.SpeechSynthesis.SpeechSynthesizer();

// Generate the audio stream from plain text.
SpeechSynthesisStream stream = await synth.SynthesizeTextToStreamAsync("Hello World");

// Send the stream to the media object.
mediaElement.SetSource(stream, stream.ContentType);
mediaElement.Play();
```

Cortana Concepts

Cortana is much more than a simple reminder service or interactive search interface. Cortana offers a robust and comprehensive extensibility framework that enables you to seamlessly incorporate functionality from your app or service into the Cortana experience via simple mechanisms that can be leveraged to handled even the most complex logic flows.



Cortana Logic Flow

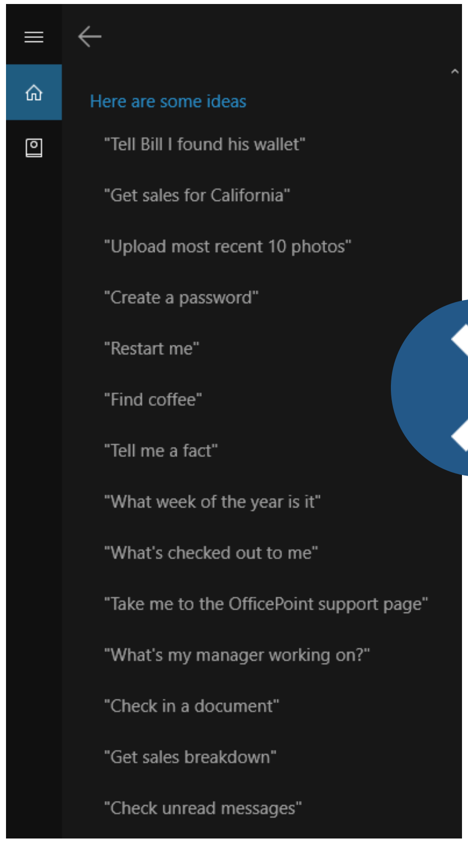

Checks to see if App exists by App Name
Instantiates App
Checks to see if App Command Set contains verbiage
Instantiates App Voice Command Service
Checks to see if App exists by Command Prefix
Instantiates App
Processes App Service logic
Processes OOTB Commands

Cortana Integration

- Update the app package manifest
- Add a background task project
- Create a command definition file
- Install a command set
- Handle voice activation
- Process interpretations
- Optionally, activate app for additional tasks

Personal Assistant Actions

Cortana interaction can trigger user actions, which can the trigger background tasks via to send information from Cortana to your app in the foreground, creating a "full circle" and loosely connected personal assistant experience.



Here are some ideas

"Tell Bill I found his wallet"

"Get sales for California"

"Upload most recent 10 photos"

"Create a password"

"Restart me"

"Find coffee"

"Tell me a fact"

"What week of the year is it"

"What's checked out to me"

"Take me to the OfficePoint support page"

"What's my manager working on?"

"Check in a document"

"Get sales breakdown"

"Check unread messages"