

Hand­s-on lab

Lab: Live Tiles and Notifications

September 2015

Contents

[Overview 3](#_Toc430692216)

[Exercise 1: Customize the Default Tile 4](#_Toc430692217)

[Task 1 – Create a blank Universal Windows app 4](#_Toc430692218)

[Task 2 – Import visual assets 6](#_Toc430692219)

[Task 3 – Update the tile badge count 8](#_Toc430692220)

[Exercise 2: Create Adaptive Live Tiles 10](#_Toc430692221)

[Task 1 – Add a model 10](#_Toc430692222)

[Task 2 – Build the tile XML 11](#_Toc430692223)

[Task 3 – Create adaptive templates for Wide and Large tiles. 14](#_Toc430692224)

[Summary 16](#_Toc430692225)

Overview

Windows 10 provides unique and engaging ways to interact with users outside of the traditional app experience. You can now display content on Live Tiles with adaptive markup, allowing you to present the best experience per tile size and screen density. Toast notifications can include interactive and adaptive elements, images, and can synchronize with your Live Tiles.

# Objectives

* 1. This lab will show you how to:
  + Set assets artwork and a background color for your default tiles
  + Use BadgeUpdateManager to update the tile badge count
  + Create adaptive templates for Live Tiles
  + Update Live Tiles
  + Support small, medium, wide, and large tiles

# System requirements

* 1. You must have the following to complete this lab:
  + Microsoft Windows 10
  + Microsoft Visual Studio 2015

# Setup

* 1. You must perform the following steps to prepare your computer for this lab:
  2. Install Microsoft Windows 10.
  3. Install Microsoft Visual Studio 2015.

# Exercises

* 1. This Hands-on lab includes the following exercises:
  2. Customize the Default Tile
  3. Launch Interactive Toast
  4. Schedule Tile Updates
  5. Estimated time to complete this lab:  **45 to 60 minutes**.

Exercise 1: Customize the Default Tile

1. Windows 10 apps use the logo assets you provide to display default tiles for your users. In this exercise, you will import logo assets to create simple tiles and update the tile badge with the BadgeUpdateManager.

Task 1 – Create a blank Universal Windows app

We will begin by creating a project from the Blank App template.

1. In a new instance of Visual Studio 2015, choose **File > New> Project** to open the New Project dialog. Navigate to **Installed > Templates > Visual C# > Windows > Universal** and select the **Blank App (Universal Windows)** template.
2. Name your project **TilesAndNotifications** and select the file system location where you will save your Hands-on Lab solutions. We have created a folder in our **C:** directory called **HOL** that you will see referenced in screenshots throughout the labs.

Leave the options selected to **Create new solution** and **Create directory for solution**. You may deselect both **Add to source control** and **Show telemetry in the Windows Dev Center** if you don't wish to version your work or use Application Insights. Click **OK** to create the project.

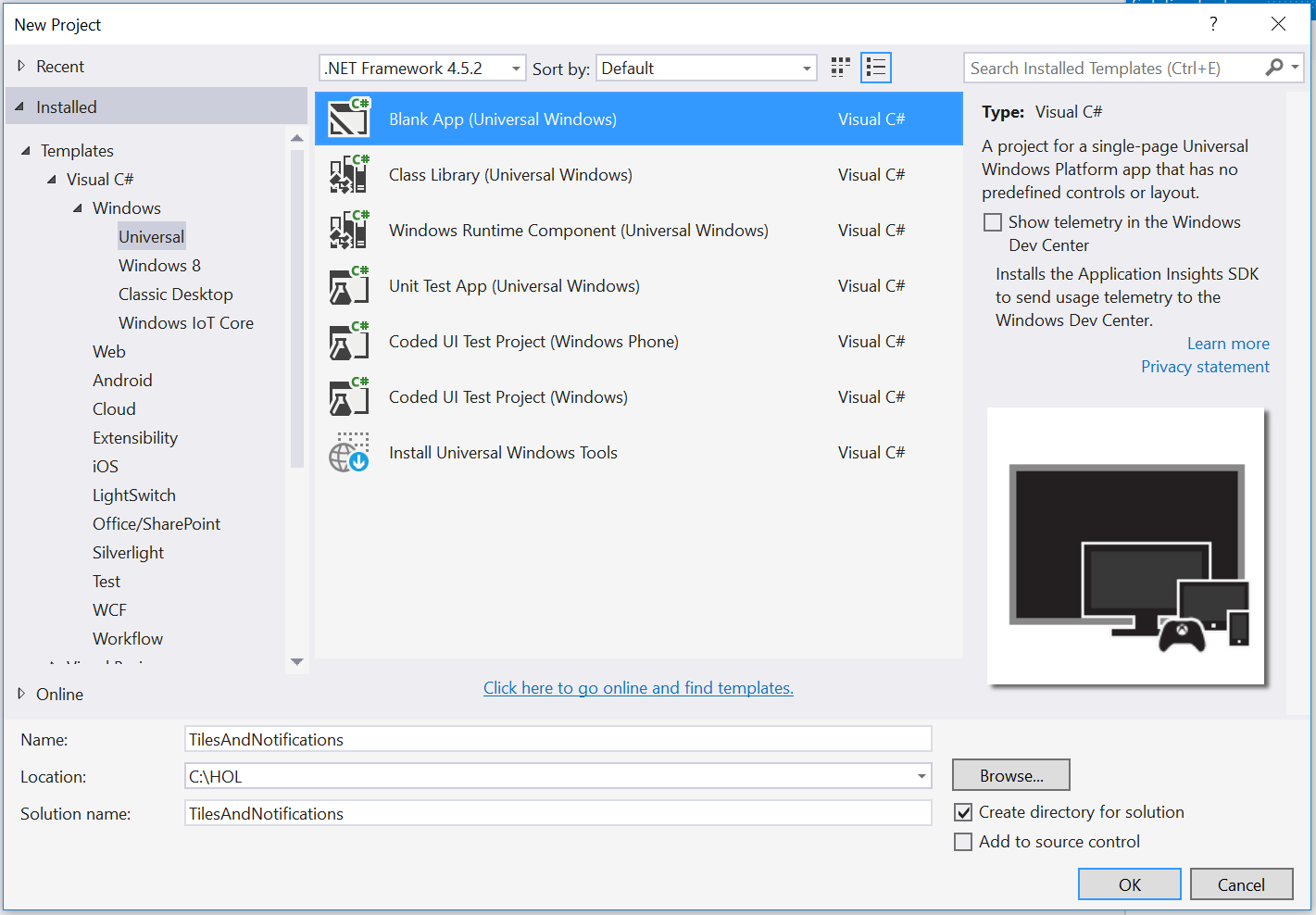


Figure 1

Create a new Blank App project in Visual Studio 2015.

1. Set your Solution Configuration to **Debug** and your Solution Platform to **x86**. Select **Local Machine** from the Debug Target dropdown menu.



Figure 2

* + 1. Configure your app to run on the Local Machine.

1. Build and run your app. You will see a blank app window with the frame rate counter enabled by default for debugging.



Figure 3

The blank universal app running in Desktop mode.

* 1. **Note:** The frame rate counter is a debug tool that helps to monitor the performance of your app. It is useful for apps that require intensive graphics processing but unnecessary for the simple apps you will be creating in the Hands-on Labs.
  2. In the Blank App template, the preprocessor directive to enable or disable the frame rate counter is in **App.xaml.cs**. The frame rate counter may overlap or hide your app content if you leave it on. For the purposes of the Hands-on Labs, you may turn it off by setting **this.DebugSettings.EnableFrameRateCounter** to **false**.

1. Return to Visual Studio and stop debugging.

Task 2 – Import visual assets

To brand your app, you can replace the generic assets from the Blank App template with your own logos. Visual assets are typically provided in a number of scale factors for different screen densities. In this task, you will add branded image assets to the app to display on default tiles.

1. Open **Package.appxmanifest** in the manifest editor and navigate to the **Visual Assets** tab. Select **Tile Images and Logos** assets in the image assets pane.
2. In the Background color field, enter **deepskyblue**.
   1. **Note:** If you leave the Background color field set to transparent, your tiles will inherit the user’s selected accent color, which the user can change in **Windows 10 Settings > Personalization > Colors**. We will set a background color for the tiles in this exercise to make it easier to see the uploaded images files in the manifest editor.
3. Use the ellipsis symbol under the **Square71x71Logo** image at **scale 200** to open the **Select Image** dialog. Navigate to your Hands-on labs **Assets** folder and select the **Square71x71Logo.scale-200.png** image file. Click **Open** to add the image as a visual asset.

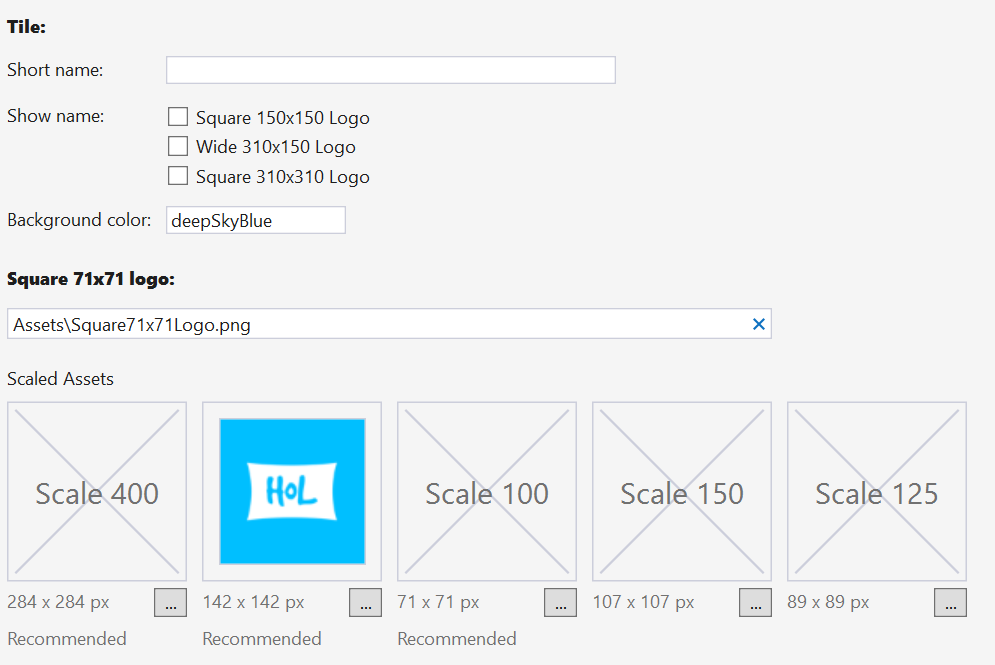


Figure 4

The Square71x71Logo asset in the manifest editor.

1. Repeat **Step 2** to add **Square150x150Logo.scale-200.png**, **Square310x310Logo.scale-200.png**, **Wide310x150Logo.scale-200.png**, and **Square44x44Logo.scale-200.png** to your visual assets.
2. Build and run your app. Once it has deployed, find your app in the Start menu. Right-click on the app name and choose **Pin to Start**.



Figure 5

Pin your app to the Start menu.

1. Your default tile will appear on the Start menu with the logo assets and background color you selected in the manifest editor. Right-click on the tile and use the **Resize** option to view your tile at different sizes.

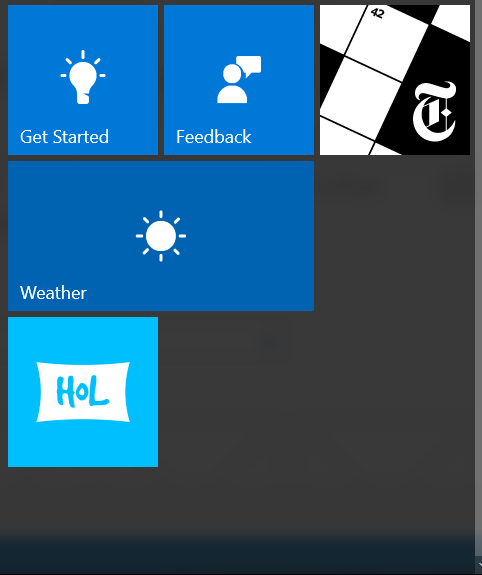


Figure 6

Pin your app to the Start menu.

1. Stop debugging and return to Visual Studio.

Task 3 – Update the tile badge count

Tile badge counts provide a great way to display information at a glance. You can update the tile badge count on the default tile without creating a Live Tile, which is a quick and easy way to add value to your app’s presence on the start screen.

1. Right-click on your project name and choose **Add > New Folder**. Name the folder **Services**.
2. Right-click on the Services folder and choose **Add > Class**. Name the class **TileService.cs**.
3. Open **TileService.cs** and make it a **public** class.
   * 1. C#
   1. namespace TilesAndNotifications.Services
   2. {
   3. public class TileService
   4. {
   5. }
4. }
5. Add a static method to update the badge count on the tile.
   * 1. C#
   1. namespace TilesAndNotifications.Services
   2. {
   3. public class TileService
   4. {
   5. static public void SetBadgeCountOnTile(int count)
   6. {
   7. // Update the badge on the real tile
   8. XmlDocument badgeXml = BadgeUpdateManager.GetTemplateContent(BadgeTemplateType.BadgeNumber);
   9. XmlElement badgeElement = (XmlElement)badgeXml.SelectSingleNode("/badge");
   10. badgeElement.SetAttribute("value", count.ToString());
   11. BadgeNotification badge = new BadgeNotification(badgeXml);
   12. BadgeUpdateManager.CreateBadgeUpdaterForApplication().Update(badge);
   13. }
   14. }
6. }
7. Return to MainPage.xaml. Create a button to update the badge count.
   * 1. XAML
   1. <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
   2. <Button Click="UpdateBadge" VerticalAlignment="Top" Margin="12">Update Badge Count</Button>
   3. </Grid>
8. In the MainPage code behind, add the **UpdateBadge()** method to call the TileService.
   * 1. C#
   1. public MainPage()
   2. {
   3. this.InitializeComponent();
   4. }
   5. private void UpdateBadge (object sender, RoutedEventArgs e)
   6. {
   7. \_count++;
   8. TileService.SetBadgeCountOnTile(\_count);
   9. }
9. Build and run your app on the Local Machine. Once it has deployed, find your app in the Start menu again. Right-click on the app name and choose **Pin to Start** if your tile does not persist .
10. 
11. Figure 7
12. Pin your app to the Start menu.
13. In your running app, click the **Update Badge Count** button. When you return to your Live Tile on the Start screen, you will see a badge count of 1. This badge count will increment every time you call the **UpdateBadge()** method.
14. Stop debugging and return to Visual Studio.

Exercise 2: Create Adaptive Live Tiles

* 1. Live Tiles can now use adaptive templates to deliver content customize to a device and screen density. In this exercise, you will create an adaptive layout for a tile. We will use static data for the tile.

Task 1 – Add a model

* 1. In a typical app, a Live Tile would display existing app data. In this task, we will create a class to mock up static data to display on your tiles.

1. Right-click on the project name and create a folder called **Models**.
2. Add a class named **PrimaryTile.cs** to the Models folder.
3. Open the **PrimaryTile** class and make it public. Add static data as string fields to populate the tile.
   * 1. C#
   1. namespace TilesAndNotifications.Models
   2. {
   3. public class PrimaryTile
   4. {
   5. public string time { get; set; } = "8:15 AM, Saturday";
   6. public string message { get; set; } = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.";
   7. public string message2 { get; set; } = " At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident.";
   8. public string branding { get; set; } = "name";
   9. public string appName { get; set; } = "HoL";
   10. }
   11. }

Task 2 – Build the tile XML

The adaptive tile schema is written in XML. In this task, you will generate the XML necessary to display text content from the PrimaryTile model on a Live Tile.

1. In **TileService.cs**, add the **TilesAndNotifications.Models** and **System.Xml.Linq** namespaces.
   * 1. C#
   1. using TilesAndNotifications.Models;
   2. using System.Xml.Linq;
2. Add a **CreateTiles** method to generate the XML for the small and medium tiles.
   * 1. C#
   1. public static Windows.Data.Xml.Dom.XmlDocument CreateTiles (PrimaryTile primaryTile)
   2. {
   3. XDocument xDoc = new XDocument(
   4. new XElement("tile", new XAttribute("version", 3),
   5. new XElement("visual",
   6. // Small Tile
   7. new XElement("binding", new XAttribute("branding", primaryTile.branding), new XAttribute("displayName", primaryTile.appName), new XAttribute("template", "TileSmall"),
   8. new XElement("group",
   9. new XElement("subgroup",
   10. new XElement("text", primaryTile.time, new XAttribute("hint-style", "caption")),
   11. new XElement("text", primaryTile.message, new XAttribute("hint-style", "captionsubtle"), new XAttribute("hint-wrap", true), new XAttribute("hint-maxLines", 3))
   12. )
   13. )
   14. ),
   15. // Medium Tile
   16. new XElement("binding", new XAttribute("branding", primaryTile.branding), new XAttribute("displayName", primaryTile.appName), new XAttribute("template", "TileMedium"),
   17. new XElement("group",
   18. new XElement("subgroup",
   19. new XElement("text", primaryTile.time, new XAttribute("hint-style", "caption")),
   20. new XElement("text", primaryTile.message, new XAttribute("hint-style", "captionsubtle"), new XAttribute("hint-wrap", true), new XAttribute("hint-maxLines", 3))
   21. )
   22. )
   23. )
   24. )
   25. )
   26. );
   27. Windows.Data.Xml.Dom.XmlDocument xmlDoc = new Windows.Data.Xml.Dom.XmlDocument();
   28. xmlDoc.LoadXml(xDoc.ToString());
   29. return xmlDoc;
   30. }
   31. **Note:** There are a number of elements you can include in your adaptive tile schema. You may choose among preset styles for each element type. For more information, visit <https://msdn.microsoft.com/en-us/library/windows/apps/Mt186446.aspx>
3. Open MainPage.xaml. Add a button below the Update Badge Count button to update the primary tile. We will enclose the buttons in a StackPanel to facilitate layout.
   * 1. XAML
4. <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
5. <StackPanel>
6. <Button Click="UpdateBadge" VerticalAlignment="Top" Margin="12">Update Badge Count</Button>
7. <Button Click="UpdatePrimaryTile" VerticalAlignment="Top" Margin="12">Update Primary Tile</Button>
8. </StackPanel>
9. </Grid>
10. In the MainPage code-behind, add the **TilesAndNotifications.Models** and **Windows.UI.Notificiations** namespaces.
    * 1. C#
    1. using TilesAndNotifications.Models;
    2. using Windows.UI.Notifications;
11. Add the **UpdatePrimaryTile()** method to the MainPage code-behind.
    * 1. C#
    1. private void UpdatePrimaryTile(object sender, Windows.UI.Xaml.RoutedEventArgs e)
    2. {
    3. var xmlDoc = TileService.CreateTiles(new PrimaryTile());
    4. var updater = TileUpdateManager.CreateTileUpdaterForApplication();
    5. TileNotification notification = new TileNotification(xmlDoc);
    6. updater.Update(notification);
    7. }
    8. **Note:** Every time you update a tile, you are actually creating a new instance of that tile. When creating the tile in an app with live data, you can pass in the latest data that you want to display on the tile.
12. Build and run your app. Repin the primary tile for your app to the Start menu if necessary.
13. Use the **Update Primary Tile** button in your running app to trigger the tile update. Open the Start menu on your device and resize the app tile to **Small**. You will see the default tile update as the Live Tile flips into view.

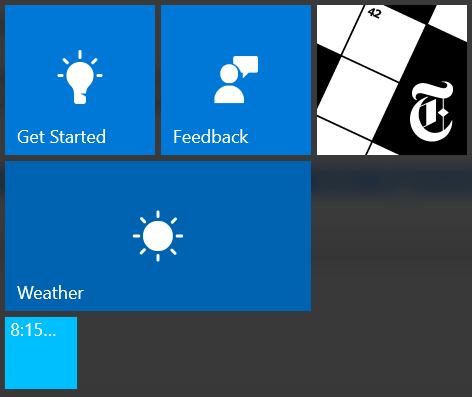


Figure 8

The small Live Tile.

1. Resize your primary tile to **Medium**. This time, when the Live Tile flips in, you will see more information provided by the adaptive template.

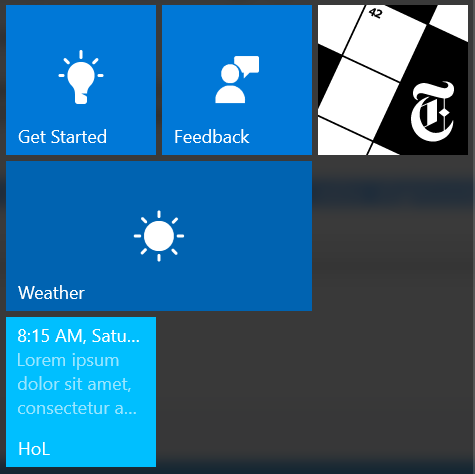


Figure 9

The medium Live Tile.

1. Stop debugging and return to Visual Studio.

Task 3 – Create adaptive templates for Wide and Large tiles.

1. Add the Wide and Large tiles to the **CreateTiles()** method. Larger live tiles have room to display images in addition to text. In this case, we will display an image that already exists in your Assets folder. Make sure to add a comma after the Medium tile XML to continue the list.
   * 1. C#
   1. ),
   2. // Wide Tile
   3. new XElement("binding", new XAttribute("branding", primaryTile.branding), new XAttribute("displayName", primaryTile.appName), new XAttribute("template", "TileWide"),
   4. new XElement("group",
   5. new XElement("subgroup",
   6. new XElement("text", primaryTile.time, new XAttribute("hint-style", "caption")),
   7. new XElement("text", primaryTile.message, new XAttribute("hint-style", "captionsubtle"), new XAttribute("hint-wrap", true), new XAttribute("hint-maxLines", 3))
   8. ),
   9. new XElement("text", primaryTile.message2, new XAttribute("hint-style", "captionsubtle"), new XAttribute("hint-wrap", true), new XAttribute("hint-maxLines", 3))
   10. ),
   11. new XElement("subgroup", new XAttribute("hint-weight", 15),
   12. new XElement("image", new XAttribute("placement", "inline"), new XAttribute("src", "Assets/StoreLogo.png"))
   13. )
   14. )
   15. ),
   16. //Large Tile
   17. new XElement("binding", new XAttribute("branding", primaryTile.branding), new XAttribute("displayName", primaryTile.appName), new XAttribute("template", "TileLarge"),
   18. new XElement("group",
   19. new XElement("subgroup",
   20. new XElement("text", primaryTile.time, new XAttribute("hint-style", "caption")),
   21. new XElement("text", primaryTile.message, new XAttribute("hint-style", "captionsubtle"), new XAttribute("hint-wrap", true), new XAttribute("hint-maxLines", 3))
   22. ),
   23. new XElement("text", primaryTile.message2, new XAttribute("hint-style", "captionsubtle"), new XAttribute("hint-wrap", true), new XAttribute("hint-maxLines", 3))
   24. ),
   25. new XElement("subgroup", new XAttribute("hint-weight", 15),
   26. new XElement("image", new XAttribute("placement", "inline"), new XAttribute("src", "Assets/StoreLogo.png"))
   27. )
   28. )
   29. )
   30. **Note:** To display wide and large Live Tiles, you must have a WideLogo and Square310x310Logo assets defined in your app manifest. We added these assets in Exercise 1.
2. Build and run your app. Pin the primary tile and resize it to **Wide** and **Large** sizes. Notice that while both have the ability to display **message** and **message2**, the large tile is more likely to have room to display it.

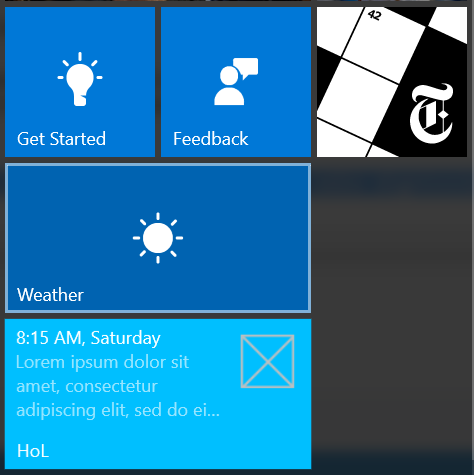


Figure 10

The wide Live Tile.

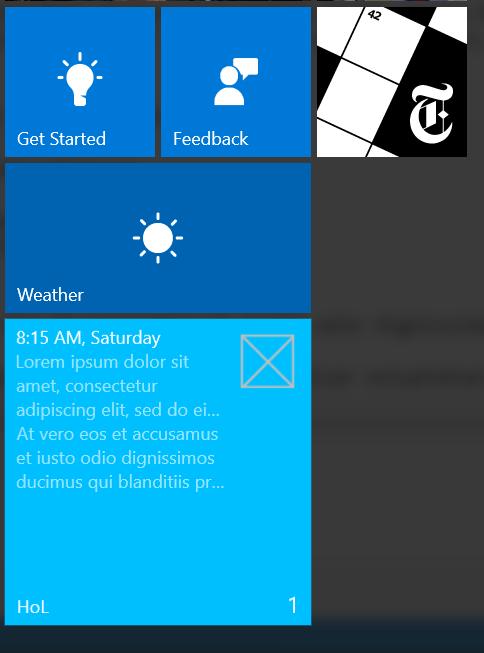


Figure 11

The large Live Tile has room to display the most content.

* 1. **Note:** Large tiles are not available on Mobile.

1. Stop debugging and return to Visual Studio.

Summary

* 1. In this lab, you branded your app with custom visual assets and used them to create a rich experience with default tiles that can display badge counts. You also learned about the new adaptive tile schema and created Live Tiles in small, medium, wide, and large sizes.