Hands-On Lab

Tiles and Notifications

Lab version: 1.0.0

Last updated: 6/20/2012



Contents

[Overview 3](#_Toc327025291)

[Exercise 1: Incorporate Secondary Tiles 4](#_Toc327025292)

[Task 1 – Modify the Application Bar 5](#_Toc327025293)

[Task 2 – Modify default.js 6](#_Toc327025294)

[Task 3 – Pin a Recipe 7](#_Toc327025295)

[Exercise 2: Incorporate Push Notifications 8](#_Toc327025296)

[Task 1 – Modify the Application Manifest 8](#_Toc327025297)

[Task 2 – Subscribe to Push Notifications 9](#_Toc327025298)

[Task 3 – Test Push Notifications 10](#_Toc327025299)

[Exercise 3: Incorporate Scheduled Toasts 11](#_Toc327025300)

[Task 1 – Modify the Application Bar 11](#_Toc327025301)

[Task 2 – Enable Toast Notifications 13](#_Toc327025302)

[Task 3 – Make a Toast! 14](#_Toc327025303)

[Summary 14](#_Toc327025304)

Overview

* 1. Tiles are an important element of a Metro style app’s user experience. When an app is installed, a tile representing it is created on the Windows 8 start screen. Known as a *primary tile*, this tile serves as a shortcut for launching the app. By default, the image for a primary tile comes from the Logo.png file included by Visual Studio in a Metro project. That image measures 150 by 150 pixels and represents a *square tile*. However, you can also include a *wide tile* in your application by adding a 310-by-150 image to your project and designating it as the “wide logo” in the application manifest. If an application supports wide tiles, users can toggle between square tiles and wide tiles on the start screen.
  2. Metro style apps also have the option of creating additional tiles known as *secondary tiles*. Secondary tiles serve as shortcuts to launch an app and have it go to a predefined location or state. For example, a weather application might allow users to create secondary tiles representing geographic locations such as Redmond, WA, and Atlanta, GA. A user may then start the application and view the weather in Redmond or Atlanta simply by tapping the secondary tile.
  3. You can create secondary tiles using WinRT’s Windows.UI.StartScreen.SecondaryTile class. Creation is usually initiated by an action performed by the user, such as tapping a command in the application bar. An application creating a secondary tile provides a number of parameters, including the URIs of the tile’s background images (both square and wide if you want the user to be able to toggle between the two) and a string representing activation arguments. When the app is launched from the secondary tile, the operating system passes the activation arguments to the app; these arguments provide the information the app needs to initialize itself. For a weather application, the activation arguments might be nothing more than a zip code.
  4. Once a tile is created, it does not have to remain static. Metro style apps enjoy a number of ways to make a tile’s content both dynamic and compelling via tile updates and push notifications. For example, an app can use the Windows.UI.Notifications.TileUpdater class to update the contents of a tile. These updates can optionally be queued; when queuing is enabled, the operating system rotates between the last five updates every few seconds, making the tile feel fresh and alive.
  5. Of course, an app has to be running to use the TileUpdater class to update a tile. But tiles can be updated even when the app isn’t running through the Windows Notification Service, or WNS. Suppose a weather app wants to display severe-weather alerts, even if it is not running. Through WNS, the servers for the weather app can send push notifications that update tiles on the start screen.
  6. Push notifications aren’t limited to updating tiles. They can also pop up a toast window containing a message for the user (for example, "Severe weather detected in your area”), and they can display a badge – a number or predefined glyph – on a tile. (Think of a mail app that wants to alert the user to new messages in his or her inbox.)

1. In this lab, you will get first-hand experience with secondary tiles, push notifications, and toasts by adding them to Contoso Cookbook. At the conclusion, users will be able to pin favorite recipes to the start screen with secondary tiles, see tiles updated by the Windows Notification Service, and see scheduled toasts in action.

# Objectives

* 1. This lab will show you how to:
  + Create secondary tiles
  + Use push notifications to update primary tiles
  + Schedule toasts to message the user even if your application isn’t running

# System Requirements

* 1. You must have the following items to complete this lab:
  + Microsoft Windows 8 Release Preview
  + Microsoft Visual Studio 2012 RC

# Setup

* 1. You must perform the following steps to prepare your computer for this lab:
  2. Install the Microsoft Windows 8 Release Preview
  3. Install the Microsoft Visual Studio 2012 RC

# Exercises

* 1. This Hands-On Lab comprises the following exercises:
  2. Incorporate Secondary Tiles
  3. Incorporate Push Notifications
  4. Incorporate Scheduled Toasts
  5. Estimated time to complete this lab: **30 to 40 minutes**.

Exercise 1: Incorporate Secondary Tiles

1. In this exercise, you’ll add a command to the application bar enabling users to pin favorite recipes to the start screen with secondary tiles. You’ll also add logic to Contoso Cookbook to show the corresponding recipe when the app is activated from a secondary tile.

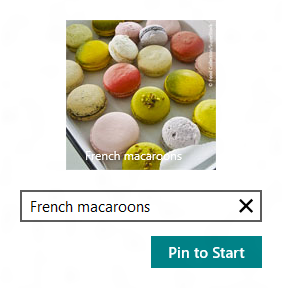
Task 1 – Modify the Application Bar

* 1. The first thing we need to do is add a Pin command to the application bar, and write a handler for it that creates a secondary tile representing the currently displayed recipe.
  2. Open the ContosoCookbook project you finished in Lab 6 in Visual Studio. If you didn’t complete Lab 6 or would like to start with a reference copy, you’ll find a completed version of the lab in the starting materials.
  3. Open itemDetail.html and find the DIV element whose ID is “appbar.”
  4. Add the following statements to the DIV after the statements that declare the Brag command:
     1. HTML
     2. <button data-win-control="WinJS.UI.AppBarCommand" data-win-options="{id:'pin', label:'Pin Recipe', icon:'pin', section: 'selection'}"></button>
  5. Open itemDetail.js and add the following statement to the others like it at the top of the file:
     1. JavaScript
     2. var start = Windows.UI.StartScreen;
  6. Now add the following statements to the end of the ready function:
     1. JavaScript
     2. // Handle click events from the Pin command
     3. document.getElementById("pin").addEventListener("click", function (e) {
     4. var uri = new Windows.Foundation.Uri("ms-appx:///" + item.tileImage);
     5. var tile = new start.SecondaryTile(
     6. item.key, // Tile ID
     7. item.shortTitle, // Tile short name
     8. item.title, // Tile display name
     9. JSON.stringify(Data.getItemReference(item)), // Activation argument
     10. start.TileOptions.showNameOnLogo, // Tile options
     11. uri // Tile logo URI
     12. );
     13. tile.requestCreateAsync();
     14. });
     15. **Note:** The tile logo URIs in this step reference 150x150 image files in the Images folder. Secondary tile images must be loaded locally. They can’t be loaded remotely like ordinary images can.
  7. Launch the application and tap a recipe to go to the item-detail page.
  8. Confirm that the application bar includes a Pin command, as shown in Figure 1.
     1. 
     2. Figure 1
     3. The Pin command
  9. Return to Visual Studio and stop debugging.

Task 2 – Modify default.js

* 1. It’s easy to create a secondary tile, but you also need to recognize when the app was activated from a secondary tile and navigate to the item-detail page to display a recipe.
  2. Open default.js and find the activated event handler.
  3. Find the args.setPromise clause at the end of the handler and rewrite it as follows:
     1. JavaScript
     2. args.setPromise(WinJS.UI.processAll().then(function () {
     3. if (args.detail.arguments !== "") {
     4. nav.history.current.initialPlaceholder = true;
     5. return nav.navigate("/pages/itemDetail/itemDetail.html", { item: JSON.parse(args.detail.arguments) });
     6. }
     7. else if (nav.location) {
     8. nav.history.current.initialPlaceholder = true;
     9. return nav.navigate(nav.location, nav.state);
     10. } else {
     11. return nav.navigate(Application.navigator.home);
     12. }
     13. }));
     14. **Note:** When a Metro style app is launched, args.detail.kind tells you *why* it was launched, and args.detail.previousExecutionState tells you whether the app was terminated after its previous run. Similarly, when an app is launched from a secondary tile, args.detail.arguments provides the activation arguments, which is the string passed in the fourth parameter to SecondaryTile’s constructor. What you’re doing here is navigating to the item-detail page to show a recipe if the app activated by tapping a secondary tile.

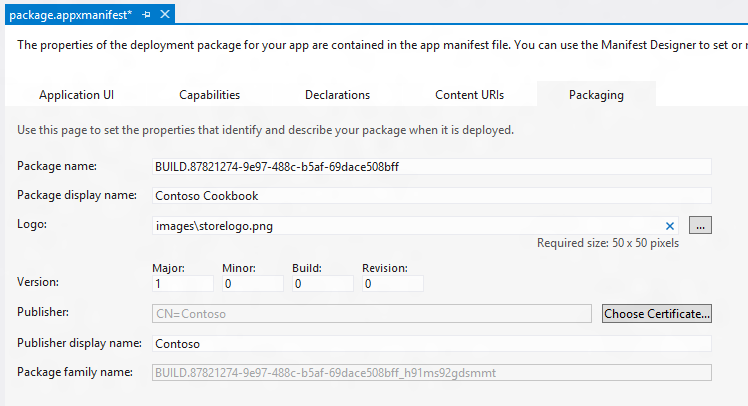
Task 3 – Pin a Recipe

* 1. Now let’s test your modifications by pinning a recipe to the start screen and making sure that the correct recipe is shown when the application is launched from the tile.
  2. Press F5 to launch the application in the debugger.
  3. Go to the item-detail page to view the recipe of your choice.
  4. Slide the application bar up from the bottom of the screen (or press Win-Z) and tap the Pin Recipe button.
  5. Tap the Pin to Start button in the ensuing popup (Figure 2).
     1. 
     2. Figure 2
     3. Pinning a secondary tile
  6. Return to Visual Studio and stop debugging.
  7. Go to the Metro start screen and verify that it now contains a secondary tile representing the recipe that you pinned.
  8. Tap the secondary tile and verify that Contoso Cookbook starts up and displays the corresponding recipe.
  9. Return to Visual Studio and stop debugging.

Exercise 2: Incorporate Push Notifications

1. Push notifications enable Metro style apps to update the content of their tiles – primary or secondary – even when the apps aren’t running. Push notifications can also update a tile’s *badge*, which is a number from 0 to 99 or a special glyph such as an asterisk that appears on the tile. Push notifications emanate from the Windows Notification Service (WNS) in response to calls from cloud-based services associated with your application. In this exercise, you’ll add push notifications to Contoso Cookbook.

Task 1 – Modify the Application Manifest

* 1. Microsoft has already deployed a Windows Azure service that Contoso Cookbook can call to subscribe to push notifications, and that transmits notifications through the WNS. When the service was deployed, it was configured to transmit notifications to applications possessing a particular package name from a particular publisher (“Contoso”). In order to receive badge notifications sent by the service, you need to modify the application manifest to use the correct package name and publisher ID.
  2. In Solution Explorer, double-click Package.appxmanifest to open the manifest.
  3. In the manifest editor, click the Packaging tab.
  4. Change Package Name to “BUILD.87821274-9e97-488c-b5af-69dace508bff”.
  5. Change Package Display Name to “Contoso Cookbook.”
  6. Change Publisher Display Name to “Contoso.”
  7. Click the “Choose Certificate” button on the Publisher line. Then select “Select from file…” from the “Configure Certificate” drop-down and select Contoso.pfx from the ensuing dialog. You’ll find Contoso.pfx in the Certs folder of the starting materials. Then click OK.
  8. Verify that Package Name, Package Display Name, Publisher, and Publisher Display Name have the values shown in Figure 3.
     1. 
     2. Figure 3
     3. The edited manifest
  9. Save your changes and close package.appxmanifest.

Task 2 – Subscribe to Push Notifications

* 1. To subscribe to push notifications, a Metro style app retrieves a notification channel from WinRT and passes the URI of the channel to a service. When the service wishes to update a tile, it calls the WNS using the channel URI, and the WNS, in turn, delivers the notification to the app. Let’s grab a channel URI when Contoso Cookbook starts up and use it to subscribe to push notifications.
  2. Open default.js and add the following statements to the others like it at the top of the file:
     1. JavaScript
     2. var notify = Windows.UI.Notifications;
     3. var push = Windows.Networking.PushNotifications;
     4. var net = Windows.Networking.Connectivity;
     5. var wsc = Windows.Security.Cryptography;
     6. var popups = Windows.UI.Popups;
  3. Find the activated event handler and add the following statements before the “if (app.sessionState.history)” statement to subscribe to push notifications each time the app starts up:
     1. JavaScript
     2. // Clear tiles and badges
     3. notify.TileUpdateManager.createTileUpdaterForApplication().clear();
     4. notify.BadgeUpdateManager.createBadgeUpdaterForApplication().clear();
     5. // Register for push notifications
     6. var profile = net.NetworkInformation.getInternetConnectionProfile();
     7. if (profile.getNetworkConnectivityLevel() === net.NetworkConnectivityLevel.internetAccess) {
     8. push.PushNotificationChannelManager.createPushNotificationChannelForApplicationAsync().then(function (channel) {
     9. var buffer = wsc.CryptographicBuffer.convertStringToBinary(channel.uri, wsc.BinaryStringEncoding.utf8);
     10. var uri = wsc.CryptographicBuffer.encodeToBase64String(buffer);
     11. WinJS.xhr({ url: "http://ContosoRecipes8.cloudapp.net?uri=" + uri + "&type=tile"}).then(function (xhr) {
     12. if (xhr.status < 200 || xhr.status >= 300) {
     13. var dialog = new popups.MessageDialog("Unable to open push notification channel");
     14. dialog.showAsync();
     15. }
     16. });
     17. });
     18. }
     19. **Note:** The call to CreatePushNotificationChannelForApplicationAsync requests a notification channel from the Metro runtime. In this example, you’re base-64-encoding the channel URI so you can pass it to the recipe service in a query string. Base-64-encoding isn’t strictly necessary, but it guards against the possibility that the channel URI might include characters that require base-64-encoding for inclusion in query strings.
     20. You’re using WinJS.xhr to fire a call off to the recipe service hosted in Windows Azure and pass the channel URI. That service maintains a record of all the clients (URIs) that have subscribed to it and fires notifications to those clients every two minutes for 20 minutes. Each channel URI passed to the service identifies a particular application running on a particular device.

Task 3 – Test Push Notifications

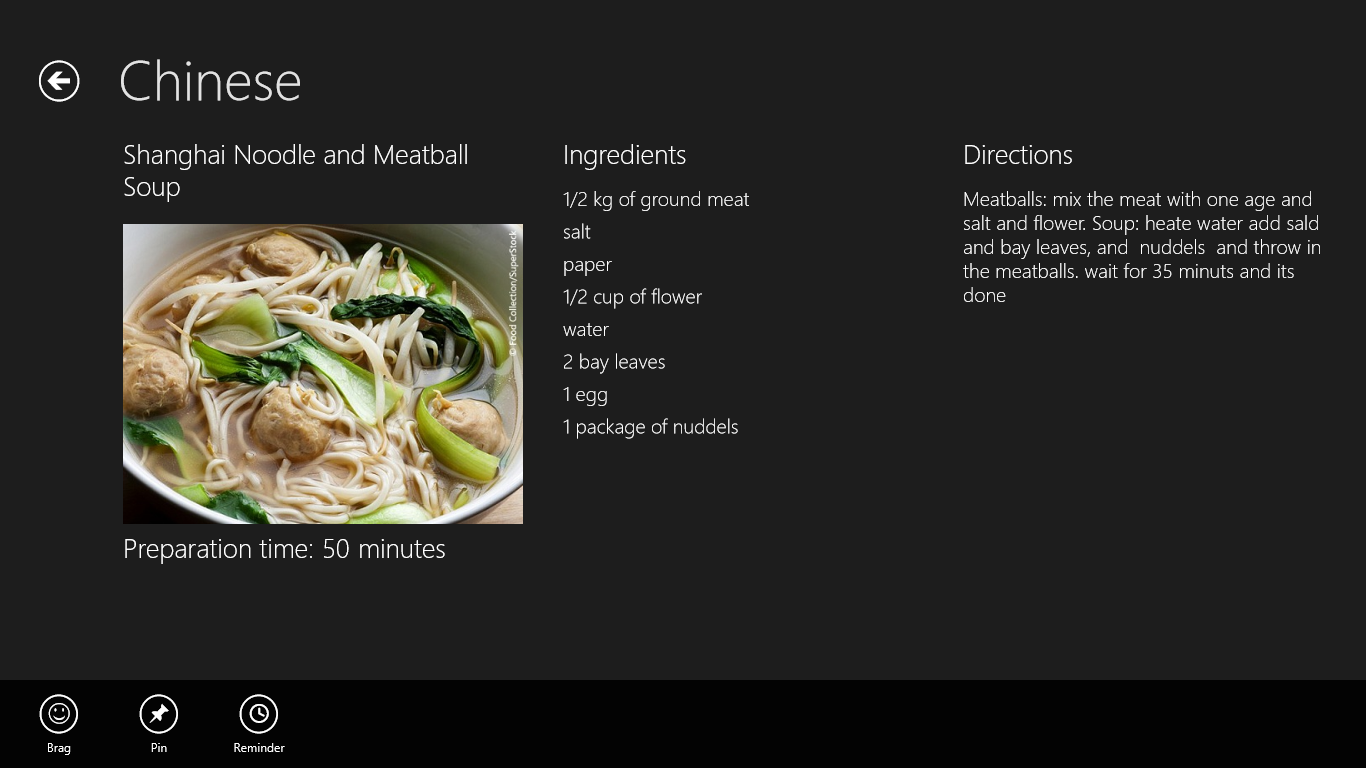
* 1. The final step is to make sure your application receives the push notifications transmitted to it.
  2. Press F5 to launch the application in the debugger. As part of its startup regimen, the application now resets its primary tile and subscribes to push notifications.
  3. Return to Visual Studio and stop debugging to close the application.
  4. Go to the Metro start screen and find the application’s primary tile. If it’s a square tile, right-click it and select “Larger” from the application bar to show a wide tile instead.
  5. Watch the tile for a few moments. Within two minutes, the tile should change to show one of several different featured recipes, as shown in Figure 4.
     1. 
     2. Figure 4
     3. Contoso Cookbook’s primary tile after a tile update
  6. Check back in another two minutes and verify that the tile has changed again. Note that all this is happening even though the application isn’t running!

**Note:** If you’d like to see badge notifications in action, change “type=tile” in the URI passed to WinJS.xhr with “type=badge”. Run the application again and after about two minutes, a number, or “badge,” will appear on the tile.

Exercise 3: Incorporate Scheduled Toasts

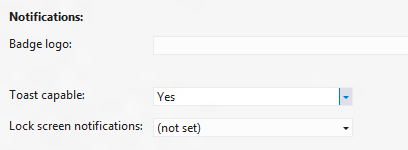
* 1. Toasts are messages that appear in “toast” windows in the corner of the screen. Toasts appear regardless of whether or not the application that scheduled them is running, and they can be scheduled by the application itself (while it’s running), or displayed in response to notifications from the Windows Notification Service. Moreover, they can include sounds as well as text.
  2. In this exercise, you’ll add a scheduled toast to Contoso Cookbook to simulate a reminder.

Task 1 – Modify the Application Bar

* 1. In order to provide a UI for scheduling a toast, we’ll add a Reminder command to the application bar and write a handler for it.
  2. Open itemDetail.html and find the DIV element whose ID is “appbar.”
  3. Add the following statements to the DIV after the statement that declares the Pin command to add a Reminder command to the application bar:
     1. HTML
     2. <button data-win-control="WinJS.UI.AppBarCommand" data-win-options="{id:'remind', label:'Reminder', icon:'clock', section: 'selection'}"></button>
  4. Open itemDetail.js and add the following statements to the others like it at the top of the file:
     1. JavaScript
     2. var notify = Windows.UI.Notifications;
     3. var popups = Windows.UI.Popups;
  5. Add the following statements to the end of the ready function:
     1. JavaScript
     2. // Handle click events from the Reminder command
     3. document.getElementById("remind").addEventListener("click", function (e) {
     4. // Create a toast notifier
     5. var notifier = notify.ToastNotificationManager.createToastNotifier();
     6. // Make sure notifications are enabled
     7. if (notifier.setting != notify.NotificationSetting.enabled) {
     8. var dialog = new popups.MessageDialog("Notifications are currently disabled");
     9. dialog.showAsync();
     10. return;
     11. }
     12. // Get a toast template and insert a text node containing a message
     13. var template = notify.ToastNotificationManager.getTemplateContent(notify.ToastTemplateType.toastText01);
     14. var element = template.getElementsByTagName("text")[0];
     15. element.appendChild(template.createTextNode("Reminder!"));
     16. // Schedule the toast to appear 30 seconds from now
     17. var date = new Date(new Date().getTime() + 30000);
     18. var stn = notify.ScheduledToastNotification(template, date);
     19. notifier.addToSchedule(stn);
     20. });
  6. Launch the application and tap a recipe to go to the item-detail page.
  7. Confirm that the application bar includes a Reminder command, as shown in Figure 5.
     1. 
     2. Figure 5
     3. The Reminder command
  8. Return to Visual Studio and stop debugging.

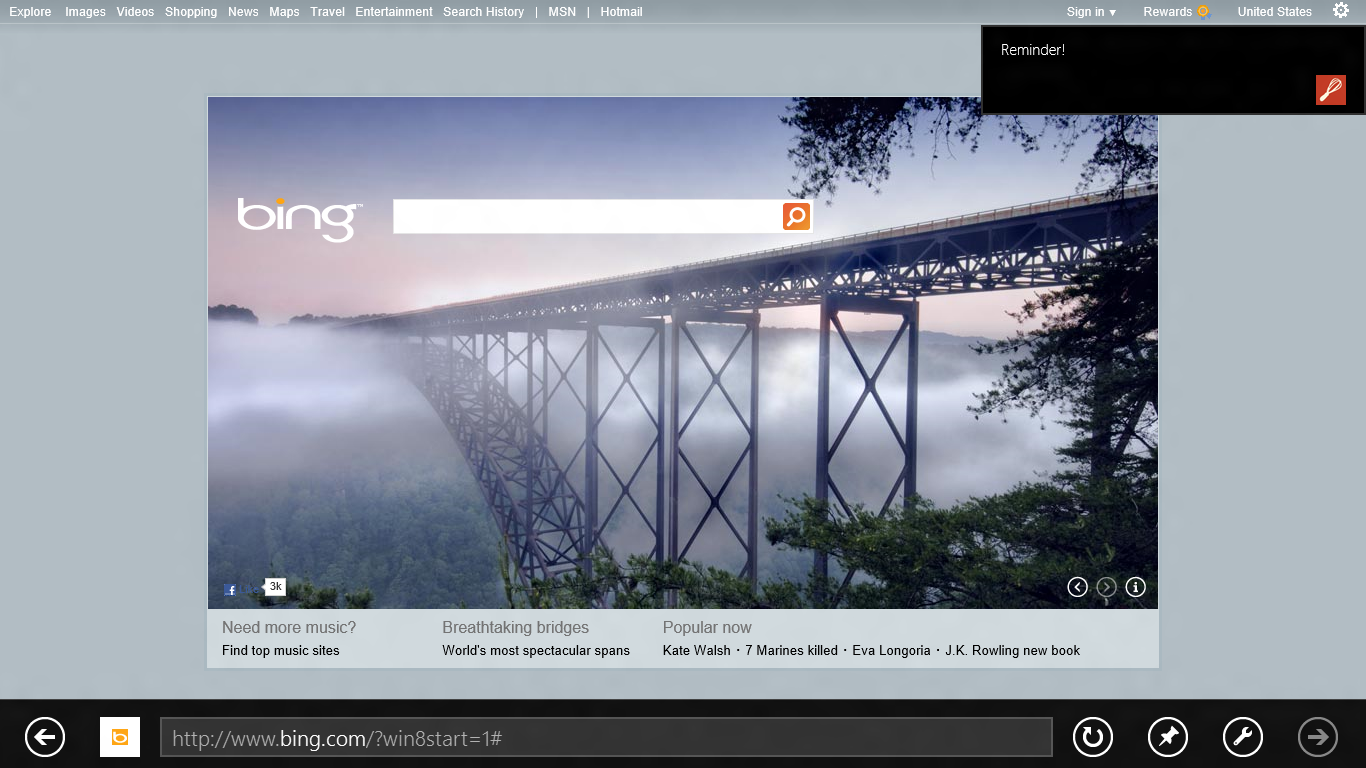
Task 2 – Enable Toast Notifications

* 1. One final task before we test is to enable toast notifications in the manifest.

1. Open Package.appxmanifest and go to the Application UI tab.
2. Change the “Toast capable” setting to “Yes” (Figure 6).
   * 1. 
     2. Figure 6
     3. Enabling toast notifications

**Note:** “Toast Capable” must be turned on before an application can schedule toasts. Once this setting is enabled, a toggle button appears in the app’s permissions page allowing the user to turn notifications on and off. The OnRedminderButtonClicked method you added checks to see if notifications are enabled and warns the user if they’re not.

Task 3 – Make a Toast!

* 1. The final task is to test the code that schedules a toast notification and see a toast in action.
  2. Press F5 to launch the application.
  3. Go to the recipe of your choice.
  4. Display the application bar and tap the “Reminder” button.
  5. Return to Visual Studio and stop debugging.
  6. Switch to the Metro start screen or to another app and wait approximately 30 seconds for a toast to appear (Figure 7).
     1. 
     2. Figure 7
     3. A toast from Contoso Cookbook
  7. Tap the toast window and verify that you switch back to Contoso Cookbook.
  8. Return to Visual Studio and stop debugging.

Summary

* 1. You’ve now seen secondary tiles, push notifications, and scheduled toasts in action. These are but a few examples of what you can do with tiles and notifications to update content when your app isn’t running.
  2. We’ve come a long way since we started on Contoso Cookbook back in Lab 1, but there’s more to do. Next up: Exploring the Windows Store APIs and allowing a user to simulate a purchase of the application and of additional recipes. It’s extremely easy to monetize a Metro application, with Microsoft doing the bulk of the work to advertise it and collect purchase fees. Sound intriguing? Then let’s get to it!