Microsoft South Africa

Hands on Lab

Contents

Part 1: Hands on Lab

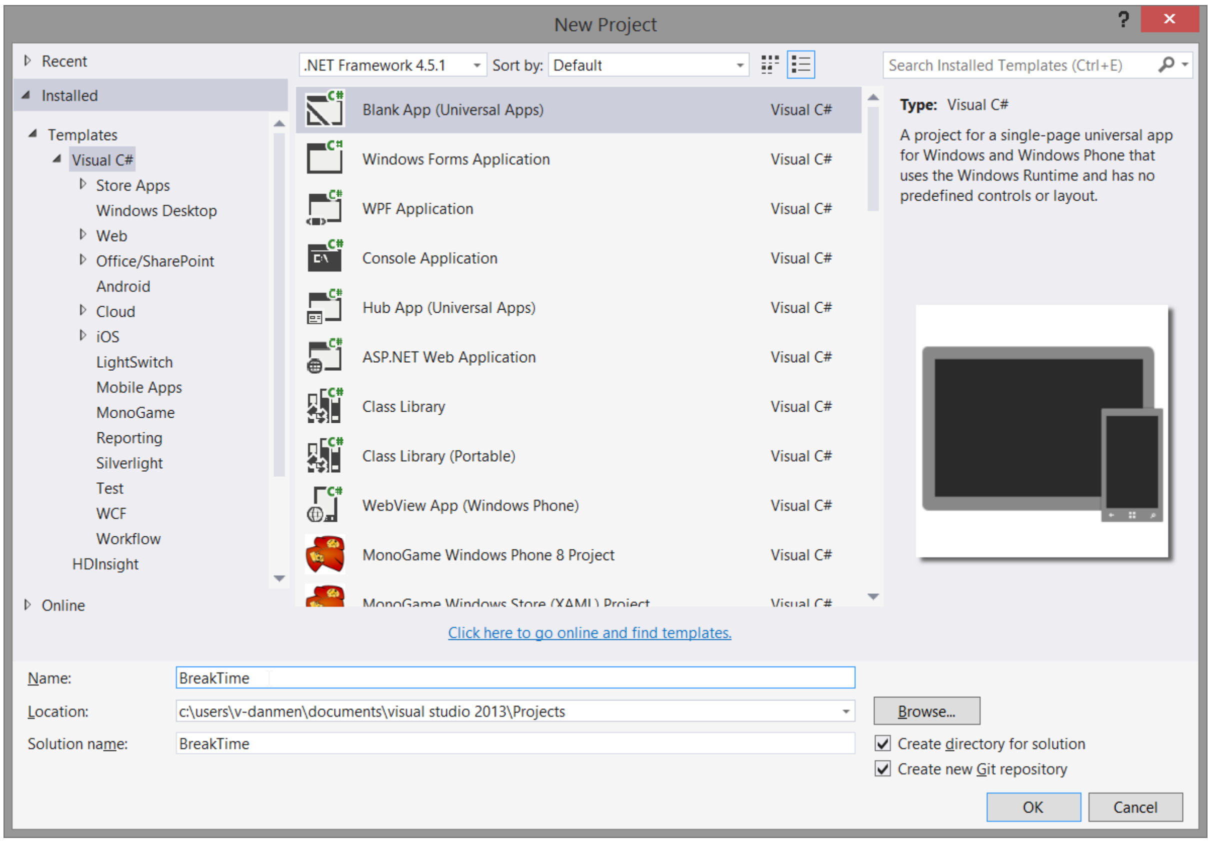
Building your first App for Windows 8.1 and publishing it to the Windows Store

The sample throughout this manual will be a break timer app for students to set times between 2 break periods and a home time and display a countdown timer to the next break and until home time. We call this app "BreakTime" App, but your app should be personal, tailored to a specific problem you want to address.

Project Setup

Create a new project by either going to "File > New > Project", or by clicking "New Project" on the Visual Studio start page.

Select "Blank App (Universal)" and change the project name to "BreakTime". Uncheck "Add to source control/create git repository" if you want. Then click OK to create the project.



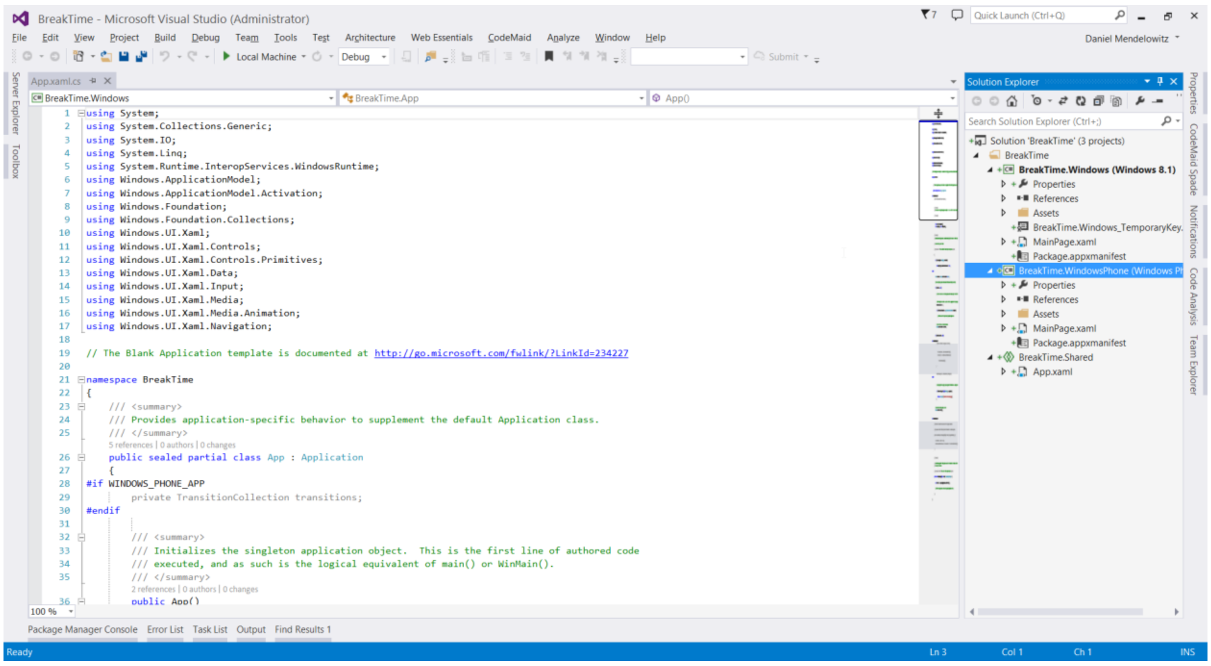
Template	Language
Blank App (Universal Apps)	Visual C#
Windows Forms Application	Visual C#
WPF Application	Visual C#
Console Application	Visual C#
Hub App (Universal Apps)	Visual C#
ASP.NET Web Application	Visual C#
Class Library	Visual C#
Class Library (Portable)	Visual C#
WebView App (Windows Phone)	Visual C#
MonoGame Windows Phone 8 Project	Visual C#
MonoGame Windows Store (XAML) Project	Visual C#

Type: Visual C#

A project for a single-page universal app for Windows and Windows Phone that uses the Windows Runtime and has no predefined controls or layout.

Name: BreakTime
Location: c:\users\v-danmen\documents\visual studio 2013\Projects
Solution name: BreakTime
☑ Create directory for solution
☑ Create new Git repository

After Visual Studio finishes initializing the project, you should be presented with App.xaml.cs open in a new tab. On the side, you should see a tab labeled "Solution Explorer". If not, go to "View > Solution Explorer" in the top menu.

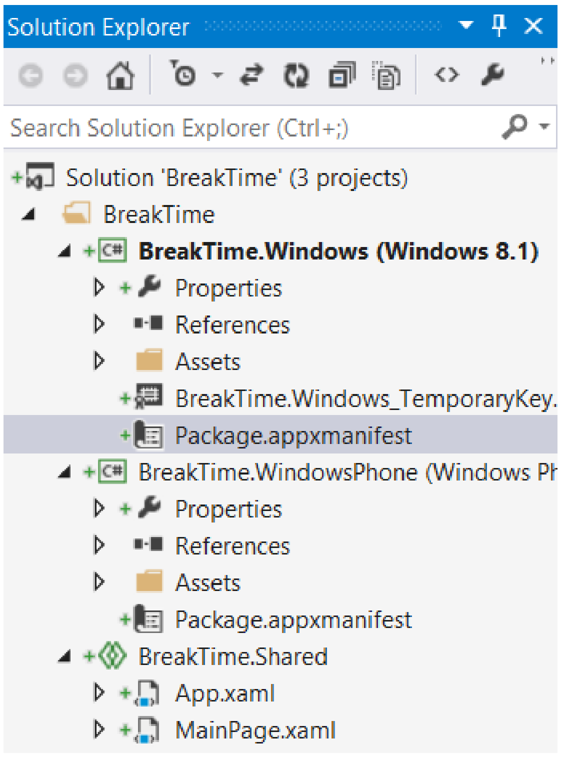


```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.ApplicationModel;
using Windows.ApplicationModel.Activation;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Media.Animation;
using Windows.UI.Xaml.Navigation;

// The Blank Application template is documented at http://go.microsoft.com/fwlink/?LinkId=234227

namespace BreakTime
{
    /// <summary>
    /// Provides application-specific behavior to supplement the default Application class.
    /// </summary>
    public sealed partial class App : Application
    {
#if WINDOWS_PHONE_APP
        private TransitionCollection transitions;
#endif

        /// <summary>
        /// Initializes the singleton application object.  This is the first line of authored code
        /// executed, and as such is the logical equivalent of main() or WinMain().
        /// </summary>
        public App()
```

Setup the Main Page

Firstly, we will move one of the MainPage.xaml files into the BreakTime.Shared namespace. You can copy either the .Windows or .WindowsPhone MainPage files as they are both identical. Once MainPage.xaml is under the ".Shared" NameSpace, delete the MainPage files under both "Windows" and "WindowsPhone" namespaces, otherwise the project will not build.

Your Solution Explorer should look like this:

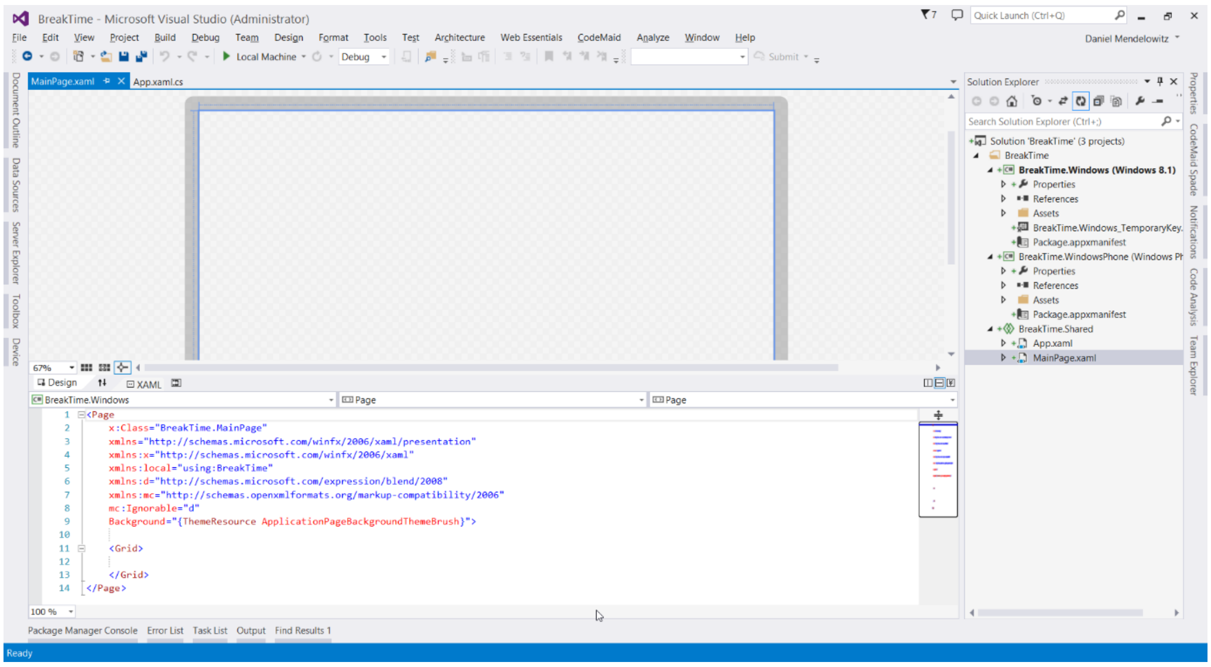


```
Solution 'BreakTime' (3 projects)
  BreakTime
    BreakTime.Windows (Windows 8.1)
      Properties
      References
      Assets
      BreakTime.Windows_TemporaryKey.
      Package.appxmanifest
    BreakTime.WindowsPhone (Windows Ph
      Properties
      References
      Assets
      Package.appxmanifest
    BreakTime.Shared
      App.xaml
      MainPage.xaml
```

By placing MainPage.xaml in the Shared Namespace, we can create the GUI and write the code only once for both Phone and Desktop.

Creating the Main GUI

Once MainPage.xaml is moved, open it by double-clicking on it. This should load a split view showing the XAML code next to a Designer preview.

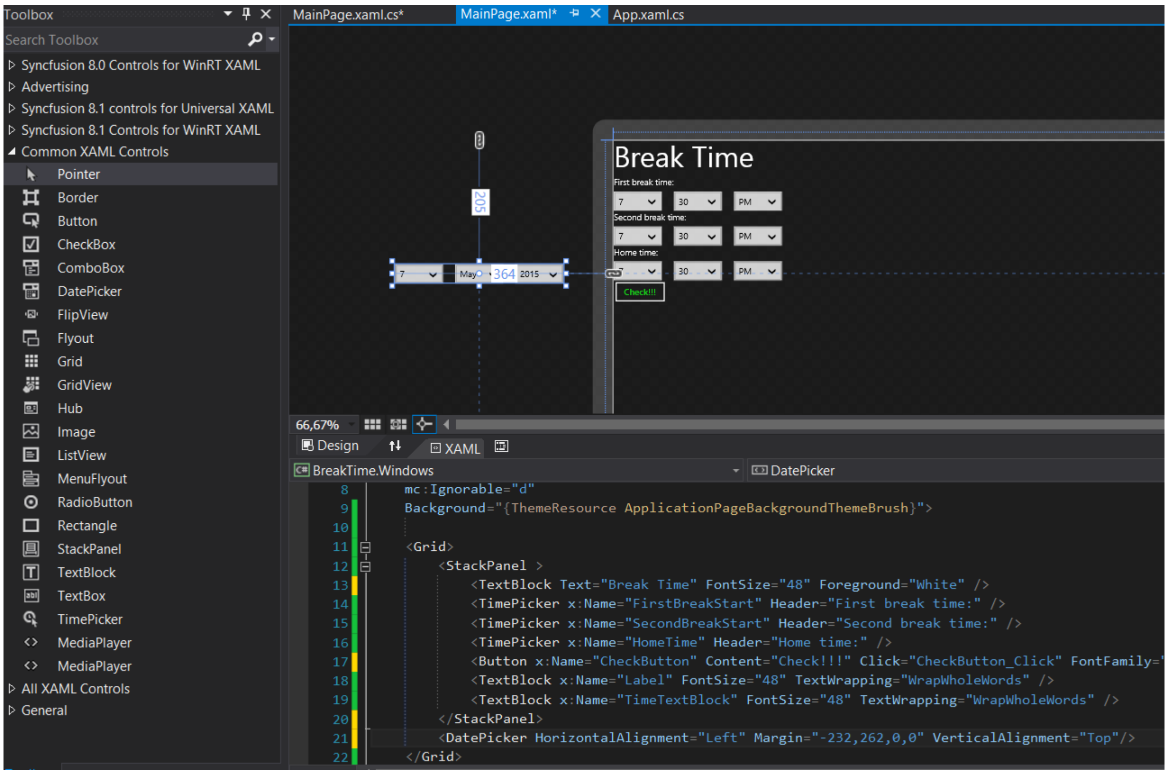


```
<Page
    x:Class="BreakTime.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:BreakTime"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid>

    </Grid>
</Page>
```

Open The Toolbox Tab (View > Toolbox) to open the controls toolbox, which contains several built in controls.

You can drag controls into the design view using the toolbox:

For this demo, we will utilize the “Button”, “DatePicker”, “StackPanel” and “Text Block” controls.



```
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid>
        <StackPanel >
            <TextBlock Text="Break Time" FontSize="48" Foreground="White" />
            <TimePicker x:Name="FirstBreakStart" Header="First break time:" />
            <TimePicker x:Name="SecondBreakStart" Header="Second break time:" />
            <TimePicker x:Name="HomeTime" Header="Home time:" />
            <Button x:Name="CheckButton" Content="Check!!!" Click="CheckButton_Click" FontFamily="
            <TextBlock x:Name="Label" FontSize="48" TextWrapping="WrapWholeWords" />
            <TextBlock x:Name="TimeTextBlock" FontSize="48" TextWrapping="WrapWholeWords" />
        </StackPanel>
        <DatePicker HorizontalAlignment="Left" Margin="-232,262,0,0" VerticalAlignment="Top"/>
    </Grid>
```

You can drag elements onto the design view and position them using the position controls

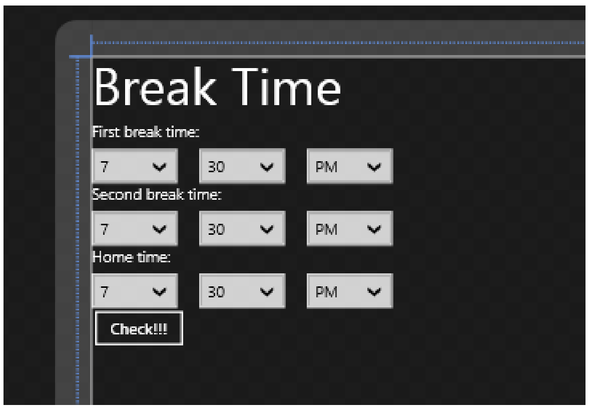
Drag the above controls into your designer window similar to as shown above. Notice that the XAML page updates as you add and manipulate controls.

You can alternatively directly edit the XAML code.

The following code is the minimal code required for the app to work.

```
<StackPanel >
       <TextBlock Text="Break Time" FontSize="48" />
       <TimePicker x:Name="FirstBreakStart" Header="First break time:" />
       <TimePicker x:Name="SecondBreakStart" Header="Second break time:" />
       <TimePicker x:Name="HomeTime" Header="Home time:" />
       <Button x:Name="CheckButton" Content="Check!!!" Click="CheckButton_Click" />
       <TextBlock x:Name="Label" FontSize="48" TextWrapping="WrapWholeWords" />
       <TextBlock x:Name="TimeTextBlock" FontSize="48" TextWrapping="WrapWholeWords"/>
</StackPanel>
```

Your designer view should now look something like this



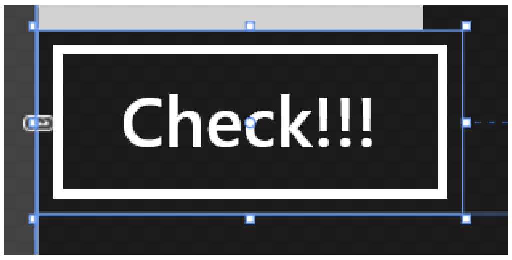
We use a <StackPanel> to layout our controls vertically without needing to drag each individual element into position.

Inside the StackPanel, We have several control types:

- The first <TextBlock> element, labelled “Break Time” has a font size of 48 and acts as a header to display the app name. You can change the font size in the properties pane or in XAML.
- We have 3 <TimePicker> Elements, each with a label for first and second break, and home time.
  The <TimePicker> allows us to set the Hour and Minute values as well as AM/PM for our break/home times.
- The <Button> named “CheckButton” which we will use to begin calculations.
- Underneath the Check button are 2 <TextBlock> elements, named “Label” and “TimeTextBlock” which we will use to display messages to the user.

We can also select elements in the designer view and edit their properties and events.

Click on the “Check!!!” button so it is highlighted like this:



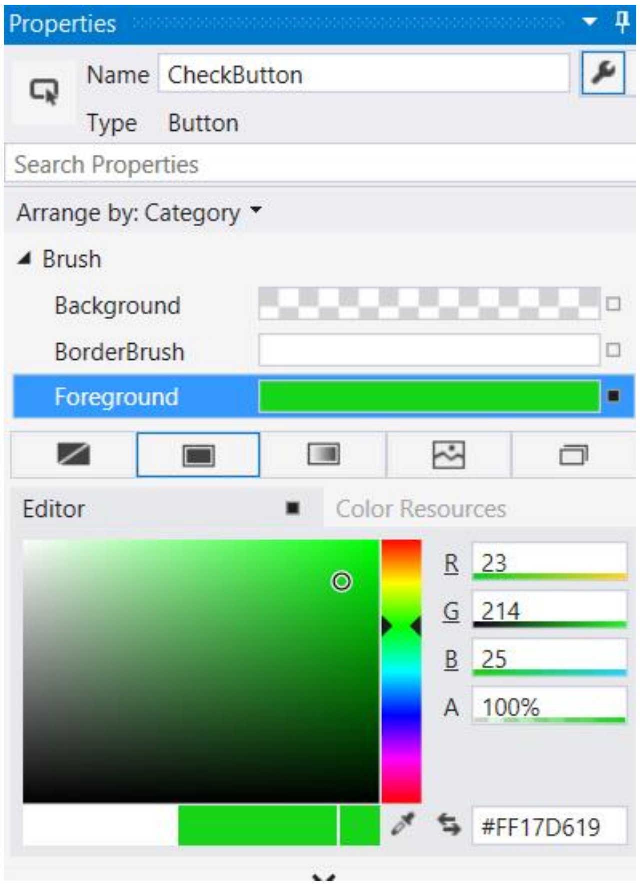
If a properties pane does not show, go to View > Properties Window

The properties pane shows all the properties that are able to be applied to the selected UI object.

You can see that the Button has a Name, “CheckButton” which corresponds to the “x:Name” property of the Button in the XAML.

Changing the properties of an element on the pane will update the values in the XAML to reflect them. Let’s change the colour of the button. In the properties view, click the “Brush” dropdown. This will open the brush editor.

Change the foreground colour of the button using a colour picker. You can also make a gradient between two colours or even use an image as a background



Properties

Name CheckButton

Type Button

Search Properties

Arrange by: Category

Brush

Background

BorderBrush

Foreground

Editor

Color Resources

R 23

G 214

B 25

A 100%

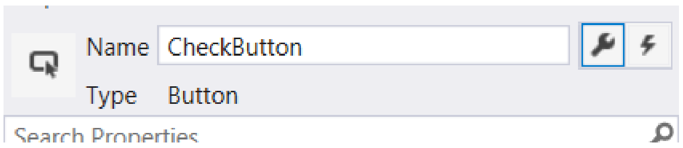
#FF17D619

Notice that when you changed the colour in properties, the <Button> tag in the XAML updated with a Foreground attribute.

Let’s make the font size of the topmost TextBlock (“Break Time”) slightly smaller. In the properties pane, open the

Adding Code

Note the Wrench and Lightning bolt icons in the top corner of the properties pane.

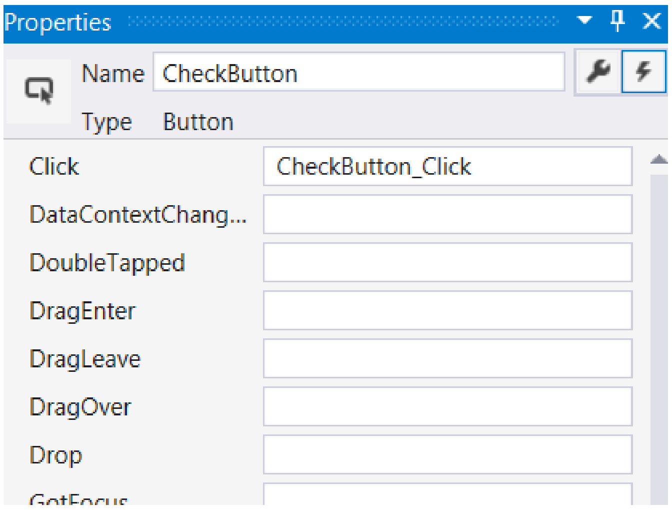


Name CheckButton

Type Button

Search Properties

Clicking the wrench icon will show the properties view, while the lightning bold switches to the events view. Open the events for the button.



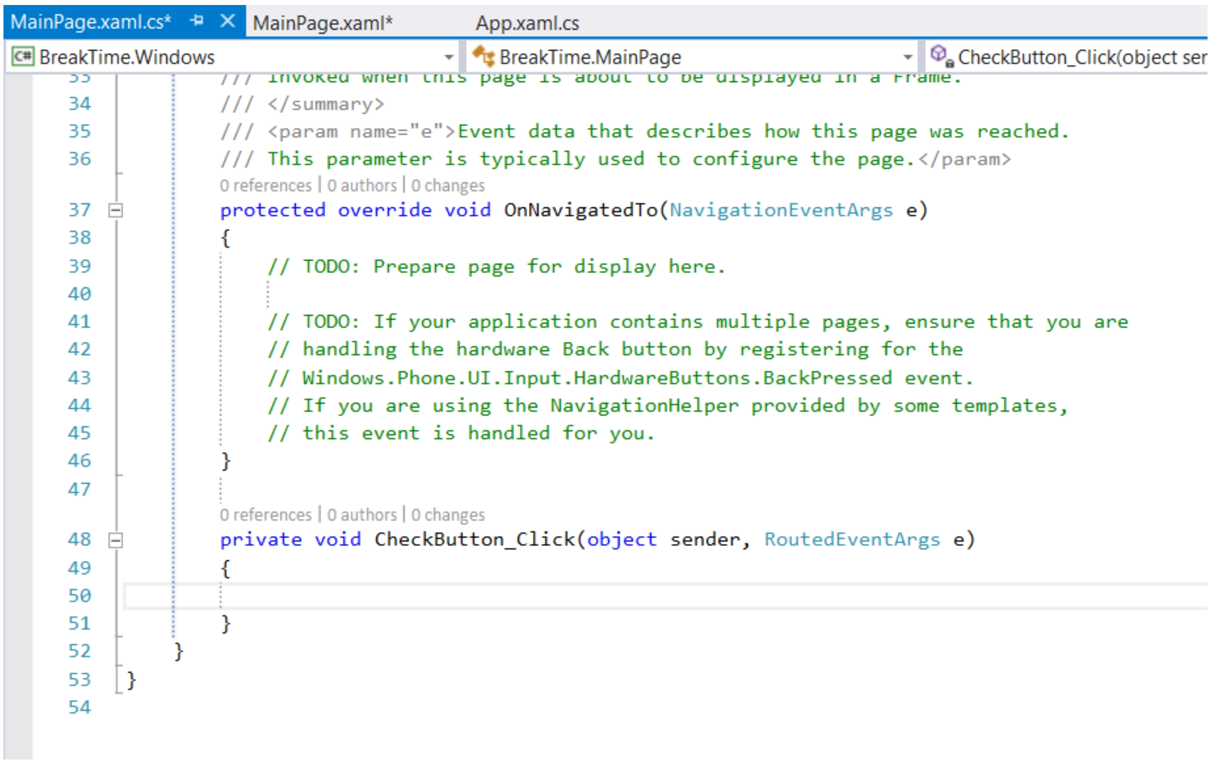
Name CheckButton

Type Button

Click	CheckButton_Click
DataContextChang...	
DoubleTapped	
DragEnter	
DragLeave	
DragOver	
Drop	
GotFocus	

We can see a list of Event names with blank textboxes with the exception of the "Click" event which has a name of "CheckButton_Click" which is the name of the method called when the button is clicked.

Double-click inside the textbox of the Click event. Doing so will create an empty CheckButton_Click event method to the MainPage.xaml.cs file. This file is called the "Code Behind" the XAML and is used to run methods on the MainPage. You can also open the code behind by right clicking on MainPage.xaml and select "View Code". The code behind will look something like this:


MainPage.xaml.cs*
MainPage.xaml*
App.xaml.cs
BreakTime.Windows
BreakTime.MainPage
CheckButton_Click(object ser
0 references | 0 authors | 0 changes
0 references | 0 authors | 0 changes

```
        /// invoked when this page is about to be displayed in a Frame.
        /// </summary>
        /// <param name="e">Event data that describes how this page was reached.
        /// This parameter is typically used to configure the page.</param>
        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            // TODO: Prepare page for display here.

            // TODO: If your application contains multiple pages, ensure that you are
            // handling the hardware Back button by registering for the
            // Windows.Phone.UI.Input.HardwareButtons.BackPressed event.
            // If you are using the NavigationHelper provided by some templates,
            // this event is handled for you.
        }

        private void CheckButton_Click(object sender, RoutedEventArgs e)
        {

        }
    }
}
```

It’s time to write some logic for the app.

The _Click method is currently empty. We will come back to it after we write some other initialisation code. Scroll to the top of the code file, and above the public MainPage() method, type the following code.

```
public sealed partial class MainPage : Page
{
    //private instance fields
    private DispatcherTimer timer = new DispatcherTimer();
    private TimeSpan firstBreakStartTime;
    private TimeSpan secondBreakStartTime;
    private TimeSpan homeTime;
    private TimeSpan now;

    public MainPage()
```

DispatcherTimer timer will be used like a stopwatch to count down to break/home time.

The 4 TimeSpan variables will be used to store the times for 1st, 2nd, and home times from the TimePicker controls, as well as the current time (now).

Inside the MainPage() method, add the following code. This will initialise the variables we just declared:

```
public MainPage()
{
    this.InitializeComponent();

    this.NavigationCacheMode = NavigationCacheMode.Required;

    firstBreakStartTime = FirstBreakStart.Time;
    secondBreakStartTime = SecondBreakStart.Time;
    homeTime = HomeTime.Time;
    now = DateTime.Now.TimeOfDay;
}
```

We assign the TimeSpans to each TimePicker.Time values as well as DateTime.Now.TimeOfDay. We can now use these values to calculate the time until break/home time.

We will now create a few helper methods to perform calculations with the TimeSpan type and to show text to the user.

```
public void TimeTrimmer(TimeSpan timeDifference)
{
    if (timeDifference < TimeSpan.FromMinutes(1))
    {
        TimeTextBlock.Text = timeDifference.Seconds.ToString() + "s";
    }
    if (timeDifference < TimeSpan.FromHours(1) && timeDifference > 
TimeSpan.FromMinutes(1))
    {
        TimeTextBlock.Text = timeDifference.Minutes.ToString() + "m" + 
timeDifference.Seconds.ToString() + "s";
    }
    if (timeDifference < TimeSpan.FromDays(1) && timeDifference > 
TimeSpan.FromHours(1))
    {
        TimeTextBlock.Text = timeDifference.Hours.ToString() + "h" + 
timeDifference.Minutes.ToString() + "m" + timeDifference.Seconds.ToString() + "s";
    }
}
```

The TimeTrimmer method takes into account whether the time until break/home time is less than 1 hour or 1 minute and will only display the smallest time, instead of 0h0m30s, it will just show 30s. It will also display a message if it is exactly break or exactly home time.

The next method, Each_Tick will check how long the current time (now) is to the next time.

```
private void Each_Tick(object sender, object e)
{
    //First break
    var firstBreakStartTime = FirstBreakStart.Time;
    var firstBreakEndTime = firstBreakStartTime + TimeSpan.FromSeconds(10);

    //Second break
    var secondBreakStartTime = SecondBreakStart.Time;
    var secondBreakEndTime = secondBreakStartTime + TimeSpan.FromSeconds(10);

    //Home time
    var homeTime = HomeTime.Time;

    //Time right now
    var now = DateTime.Now.TimeOfDay;

    var timeDifference = TimeSpan.Zero;

    if (now < firstBreakStartTime)
    {
        timeDifference = firstBreakStartTime - now;
        Label.Text = "First break in...";
        TimeTrimmer(timeDifference);
    }
    if (now > firstBreakStartTime && now < firstBreakEndTime)
    {
        Label.Text = "Yay!";
        TimeTextBlock.Text = "It is now first break!!";
    }
    if (now > firstBreakEndTime && now < secondBreakStartTime)
    {
        timeDifference = secondBreakStartTime - now;
        Label.Text = "Second break in...";
        TimeTrimmer(timeDifference);
    }
    if (now > secondBreakStartTime && now < secondBreakEndTime)
    {
        Label.Text = "Yay!";
        TimeTextBlock.Text = "It is now second break!!";
    }
    if (now > secondBreakEndTime && now < homeTime)
    {
        timeDifference = homeTime - now;
        Label.Text = "Home time in...";
        TimeTrimmer(timeDifference);
    }
    if (now > homeTime)
    {
        Label.Text = "HEY!";
        TimeTextBlock.Text = "Shouldn't you be home already??";
    }
}
```

This sets our timer to call Each_Tick every second. We also want to be able to update the display each time the timer ticks. To do this, we add an event delegate to Each_Tick every time the timer.Tick function is called.

We want to call Each_Tick once per second so that our UI will update the countdown. Back in our MainPage() method, add the following:

```
public MainPage()
{
    this.InitializeComponent();

    this.NavigationCacheMode = NavigationCacheMode.Required;

    firstBreakStartTime = FirstBreakStart.Time;
    secondBreakStartTime = SecondBreakStart.Time;
    homeTime = HomeTime.Time;
    now = DateTime.Now.TimeOfDay;
    timer.Interval = new TimeSpan(0, 0, 0, 1);
    timer.Tick += Each_Tick

}
```

We are nearly done with part 1. Let's go back to the CheckButton_Click method. What we want to happen is that when the CheckButton is pressed, the timer starts and ticks once per second.

We do this by simply adding timer.Start() to the Click method:

```
private void CheckButton_Click(object sender, RoutedEventArgs e)
{
    timer.Start();
}
```

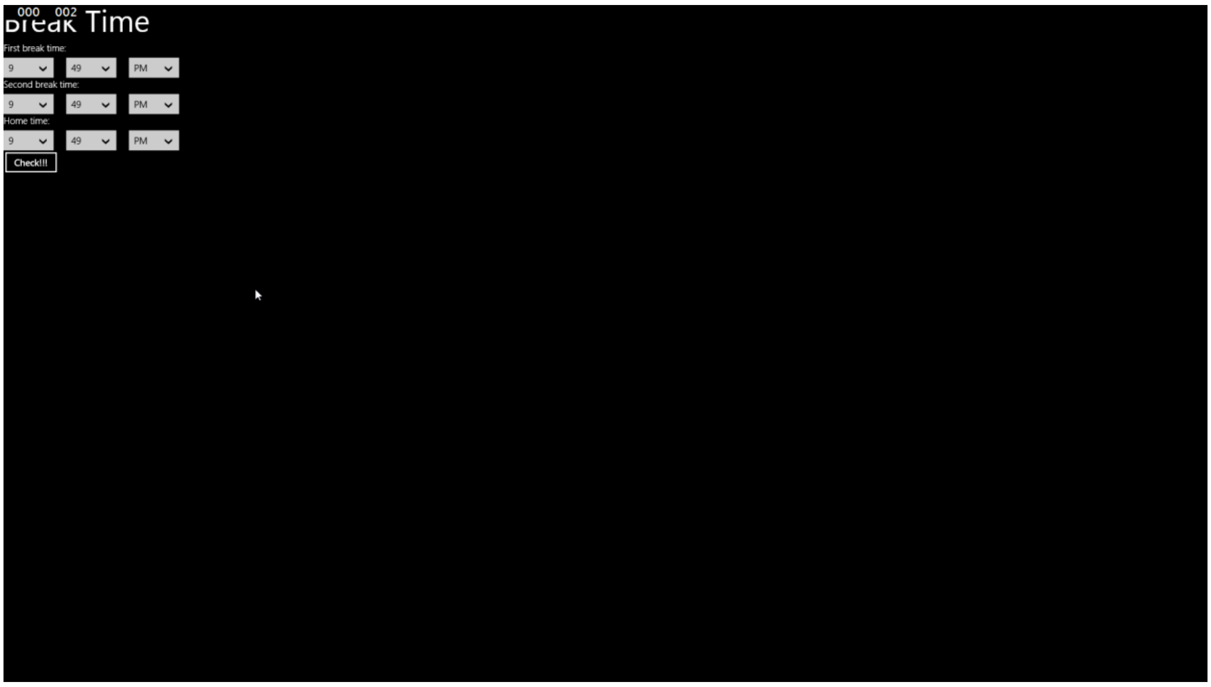
The App is nearly done!

Debugging the App

We will start by running the application on Windows 8.1. In solution explorer, right click the BreakTime.Windows Namespace and select "Set as Startup Project".

Let's run the app now. On the menu, go to Build > Build Solution, or hit F6 to build the project. Once the build completes without errors you can debug the application by pressing F5 or on the menu, Debug > Start Debugging.

The Windows app will open and you should see this:

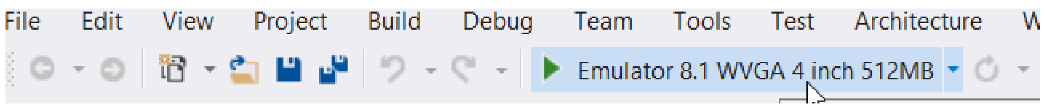


First break time: 9 49 PM
Second break time: 9 49 PM
Home time: 9 49 PM

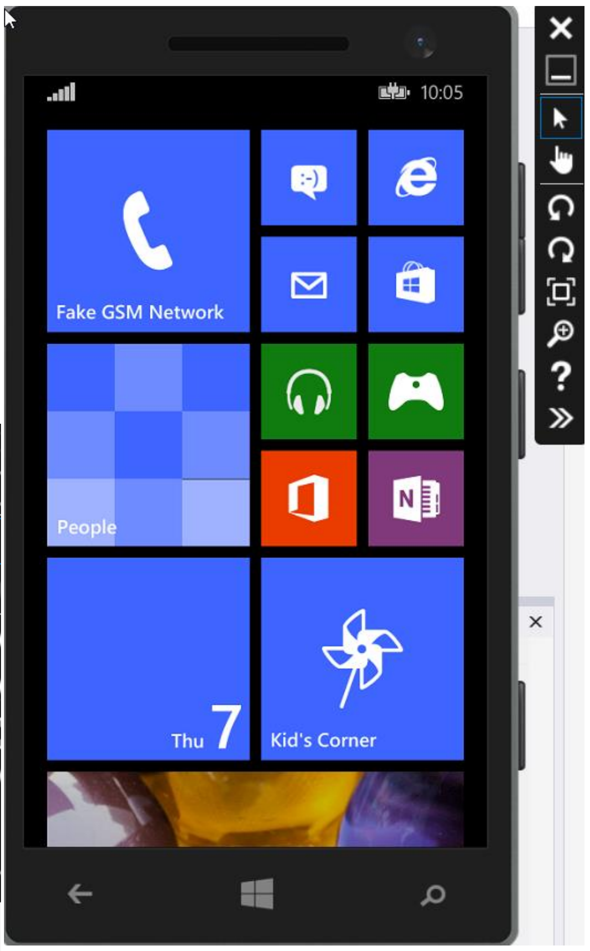
Change the values of the Time Pickers. When we click the Check!!! button, the TextBlock elements will show either a message, or a countdown to the nearest time!!!

Congratulations, you've written a simple Windows app! With Universal Apps, we can write a single UI layout that will work on Windows Phone, Windows 8, and soon, Windows 10 and Xbox One.

We can preview how the app will look on Windows Phone using the Phone Emulator, included with Visual Studio 2013 Community Edition. To run the app on the phone emulator, we must switch the startup project to BreakTime.WindowsPhone by right clicking the namespace and selecting "Set as Startup Project". On the top menu bar, there should be a green "play" button labelled "Emulator 8.1 WVGA 4 inch 512MB" if not, select it from the dropdown menu next to the button.

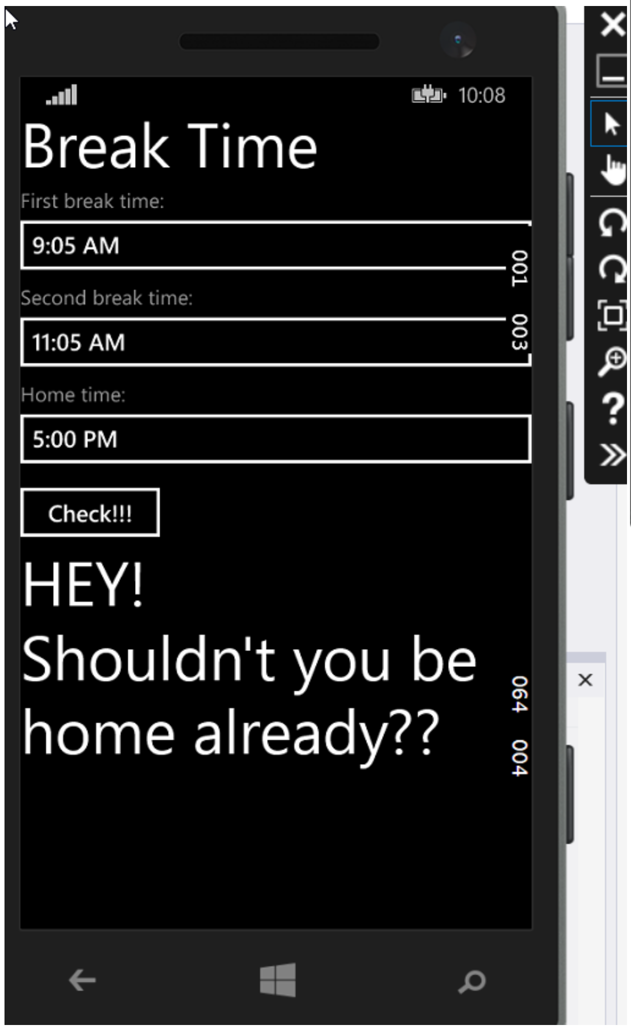


Clicking this, or pressing F5 will start the Windows Phone Emulator. The emulator takes some time to start up as it is loading a full Windows Phone 8.1 OS inside a managed virtual machine.



The emulator window will open to the start screen while Visual Studio deploys the built app to the device. When the app launches, it will show the exact same controls we saw on the Windows 8.1 app.

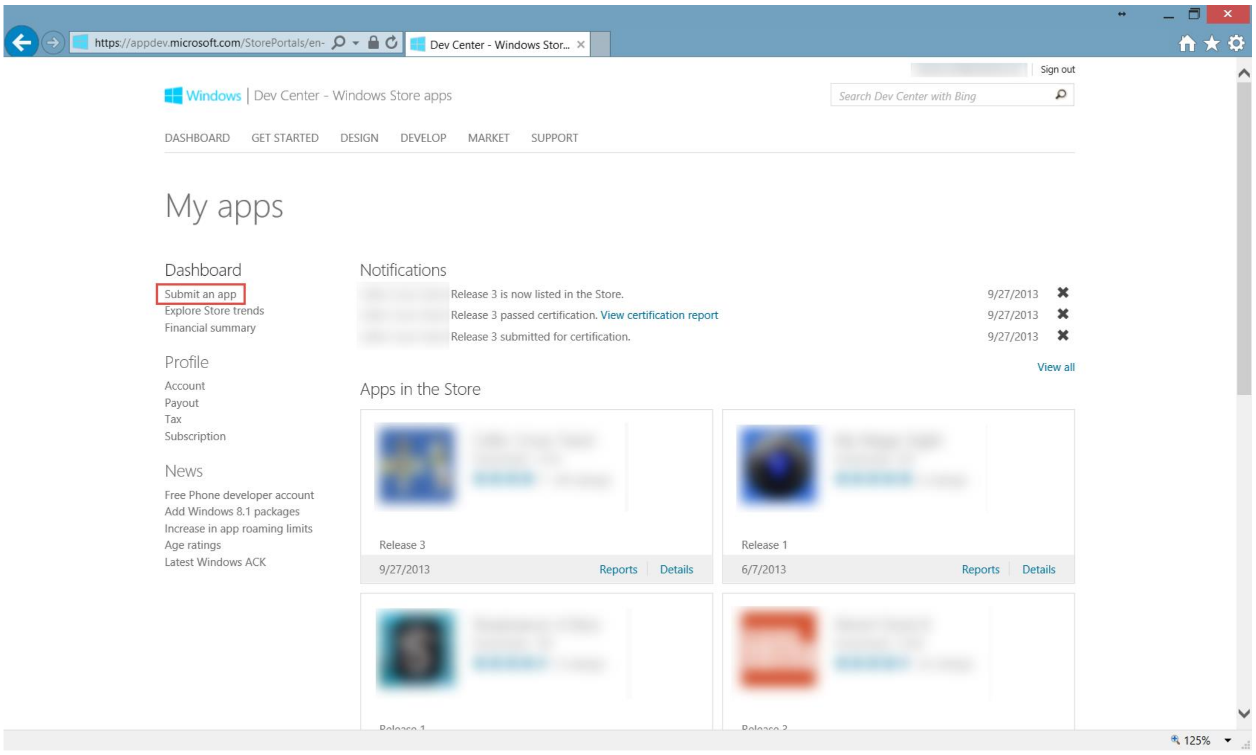
We have now created a universal app that can run on multiple platforms!



Part 2: Publishing your app to the store

Reserving the App name in the Windows Store

Head over to the Windows Developer Center http://dev.windows.com and click on the "Dashboard" link. Log in using your Microsoft ID that you have used to set up your Dev Center account. You should see a page similar to the following.



https://appdev.microsoft.com/StorePortals/en-
Dev Center - Windows Stor...
Sign out
Windows | Dev Center - Windows Store apps
Search Dev Center with Bing
DASHBOARD GET STARTED DESIGN DEVELOP MARKET SUPPORT

My apps

Dashboard
Submit an app
Explore Store trends
Financial summary

Profile
Account
Payout
Tax
Subscription

News
Free Phone developer account
Add Windows 8.1 packages
Increase in app roaming limits
Age ratings
Latest Windows ACK

Notifications

Release 3 is now listed in the Store.	9/27/2013
Release 3 passed certification. View certification report	9/27/2013
Release 3 submitted for certification.	9/27/2013

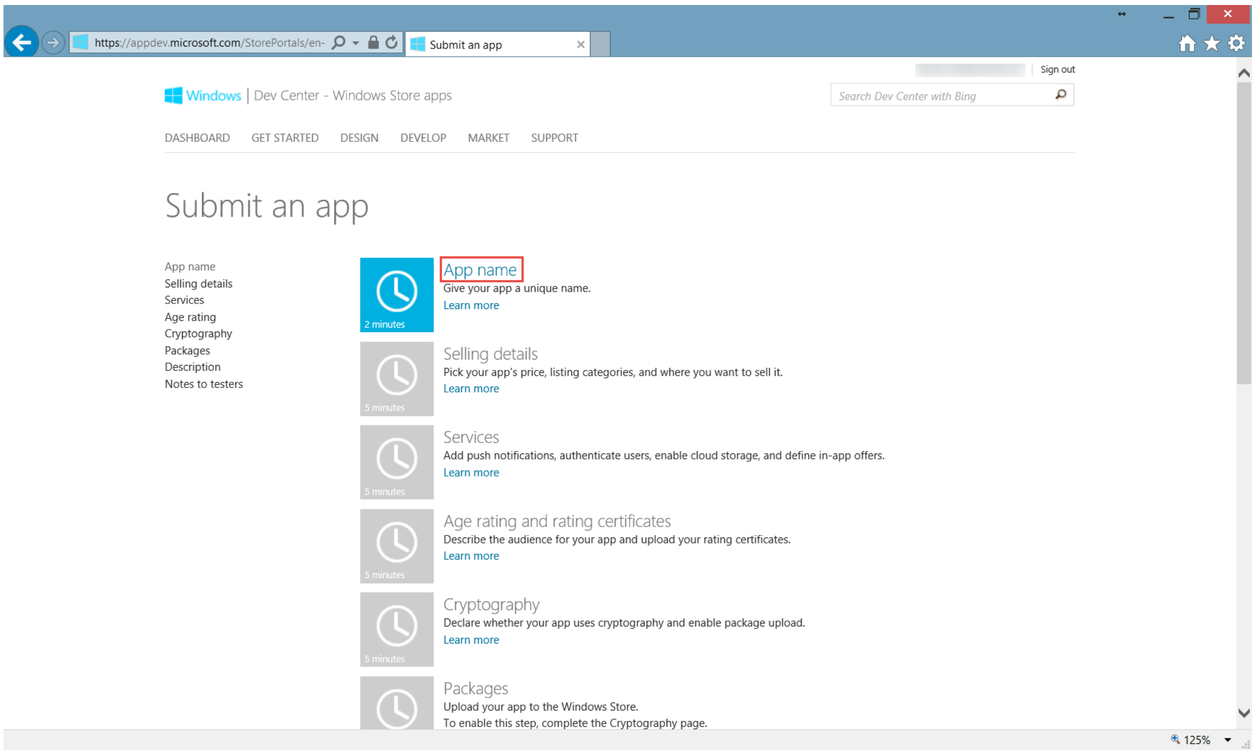
View all

Apps in the Store

Release 3 — 9/27/2013 — Reports | Details
Release 1 — 6/7/2013 — Reports | Details

125%

Click on the "Submit an app" link.



https://appdev.microsoft.com/StorePortals/en-
Submit an app
Sign out
Windows | Dev Center - Windows Store apps
Search Dev Center with Bing
DASHBOARD GET STARTED DESIGN DEVELOP MARKET SUPPORT

Submit an app

App name
Selling details
Services
Age rating
Cryptography
Packages
Description
Notes to testers

App name (2 minutes)
Give your app a unique name.
Learn more

Selling details (5 minutes)
Pick your app's price, listing categories, and where you want to sell it.
Learn more

Services (5 minutes)
Add push notifications, authenticate users, enable cloud storage, and define in-app offers.
Learn more

Age rating and rating certificates (5 minutes)
Describe the audience for your app and upload your rating certificates.
Learn more

Cryptography (5 minutes)
Declare whether your app uses cryptography and enable package upload.
Learn more

Packages
Upload your app to the Windows Store.
To enable this step, complete the Cryptography page.

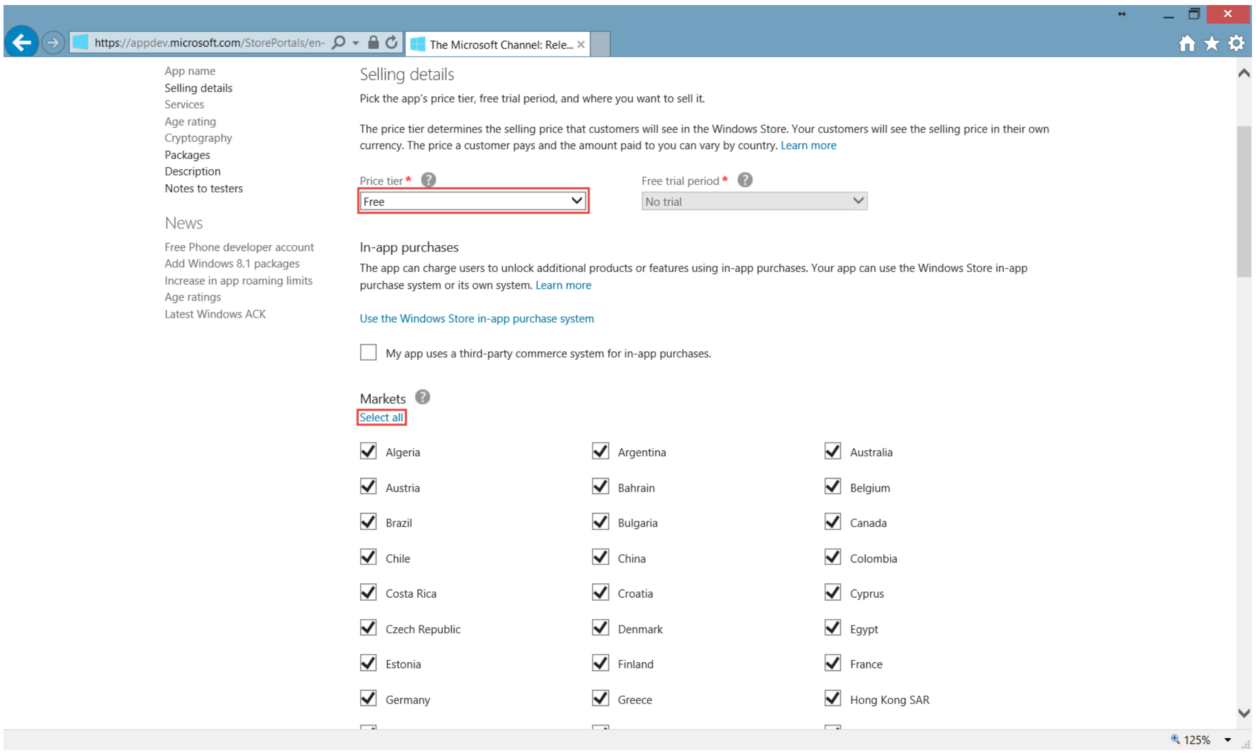
125%

First let's just reserve a name for our app. Click on "App name" and Type a name that fits/suits your own channel selection. For the test app in this manual, I chose "Break Time!".

Once you have reserved the name of your app, note it as well as we will use it later in our app.

Selling Details

Select a price tier, I recommend offering this app for "Free". As we created our app in English, it's possible to publish it to all markets around the globe. If you would like to do so, you can simply click the "Select all" link under "Markets". If the YouTube channel you chose contains any references to gambling, alcohol, inappropriate humor or other adult content, the app may not pass certification in some Asian markets. In this case, you may want to restrict the markets to which you publish.



Selling details

Pick the app's price tier, free trial period, and where you want to sell it.

The price tier determines the selling price that customers will see in the Windows Store. Your customers will see the selling price in their own currency. The price a customer pays and the amount paid to you can vary by country. Learn more

Price tier * : Free

Free trial period * : No trial

In-app purchases

The app can charge users to unlock additional products or features using in-app purchases. Your app can use the Windows Store in-app purchase system or its own system. Learn more

Use the Windows Store in-app purchase system

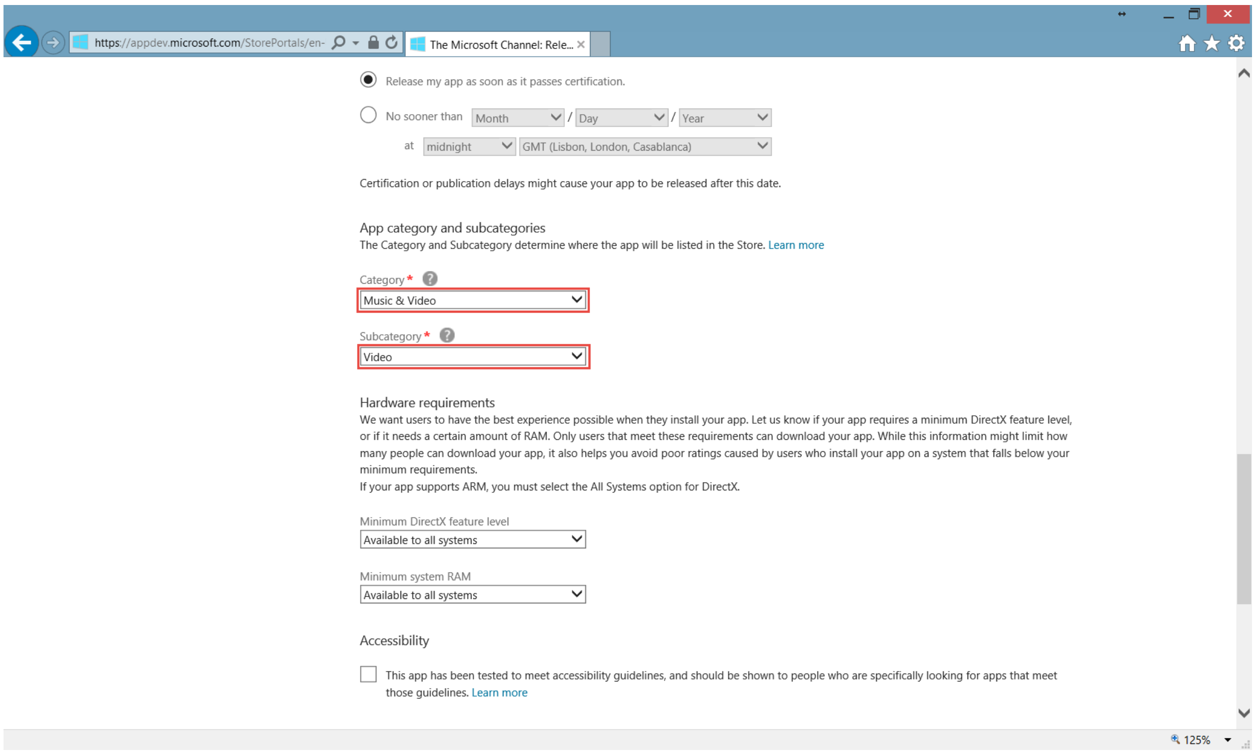
☐ My app uses a third-party commerce system for in-app purchases.

Markets

Select all

✓ Algeria	✓ Argentina	✓ Australia
✓ Austria	✓ Bahrain	✓ Belgium
✓ Brazil	✓ Bulgaria	✓ Canada
✓ Chile	✓ China	✓ Colombia
✓ Costa Rica	✓ Croatia	✓ Cyprus
✓ Czech Republic	✓ Denmark	✓ Egypt
✓ Estonia	✓ Finland	✓ France
✓ Germany	✓ Greece	✓ Hong Kong SAR

Set a category and subcategory. I chose "Music & Video" and "Video" as a good match for our app.


https://appdev.microsoft.com/StorePortals/en- | The Microsoft Channel: Rele...

● Release my app as soon as it passes certification.

○ No sooner than Month / Day / Year

at midnight GMT (Lisbon, London, Casablanca)

Certification or publication delays might cause your app to be released after this date.

App category and subcategories

The Category and Subcategory determine where the app will be listed in the Store. Learn more

Category *: Music & Video

Subcategory *: Video

Hardware requirements

We want users to have the best experience possible when they install your app. Let us know if your app requires a minimum DirectX feature level, or if it needs a certain amount of RAM. Only users that meet these requirements can download your app. While this information might limit how many people can download your app, it also helps you avoid poor ratings caused by users who install your app on a system that falls below your minimum requirements.

If your app supports ARM, you must select the All Systems option for DirectX.

Minimum DirectX feature level: Available to all systems

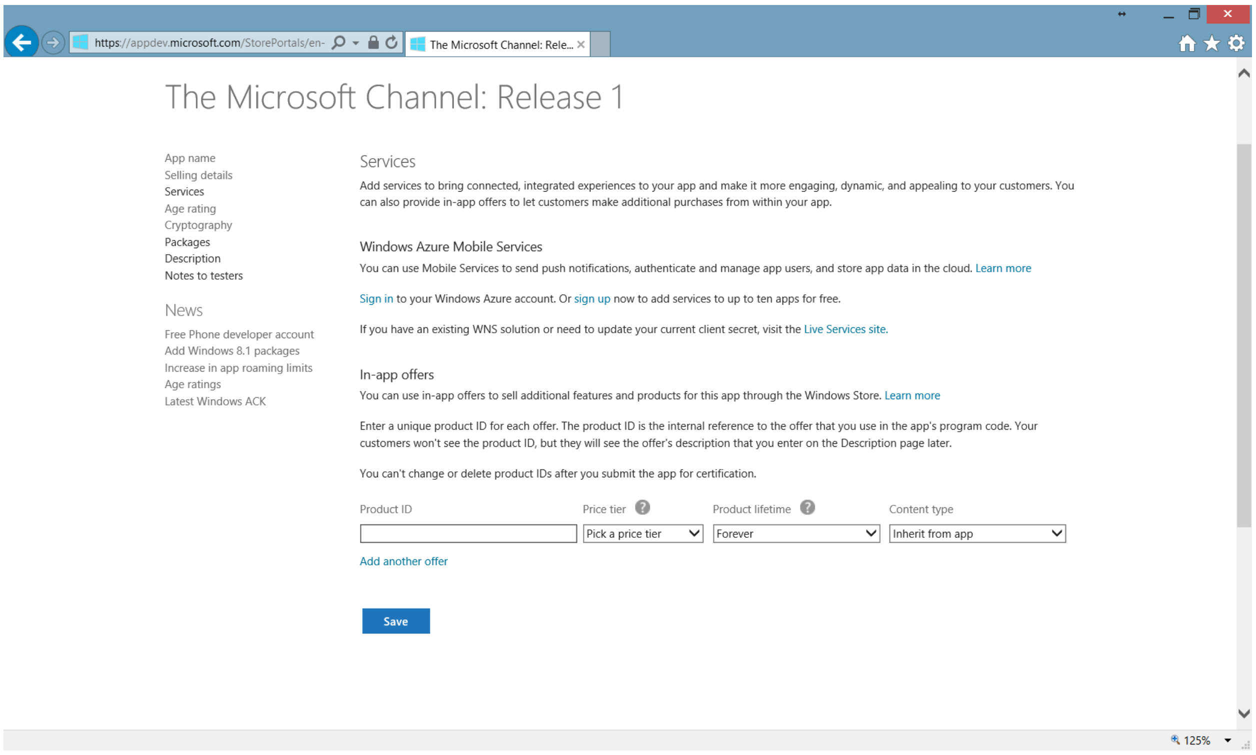
Minimum system RAM: Available to all systems

Accessibility

☐ This app has been tested to meet accessibility guidelines, and should be shown to people who are specifically looking for apps that meet those guidelines. Learn more

Services

There is no need to change anything in the “Services” page. We don’t offer any In-App-Purchases.


https://appdev.microsoft.com/StorePortals/en- | The Microsoft Channel: Rele...

The Microsoft Channel: Release 1

App name
Selling details
Services
Age rating
Cryptography
Packages
Description
Notes to testers

News

Free Phone developer account
Add Windows 8.1 packages
Increase in app roaming limits
Age ratings
Latest Windows ACK

Services

Add services to bring connected, integrated experiences to your app and make it more engaging, dynamic, and appealing to your customers. You can also provide in-app offers to let customers make additional purchases from within your app.

Windows Azure Mobile Services

You can use Mobile Services to send push notifications, authenticate and manage app users, and store app data in the cloud. Learn more

Sign in to your Windows Azure account. Or sign up now to add services to up to ten apps for free.

If you have an existing WNS solution or need to update your current client secret, visit the Live Services site.

In-app offers

You can use in-app offers to sell additional features and products for this app through the Windows Store. Learn more

Enter a unique product ID for each offer. The product ID is the internal reference to the offer that you use in the app's program code. Your customers won't see the product ID, but they will see the offer's description that you enter on the Description page later.

You can't change or delete product IDs after you submit the app for certification.

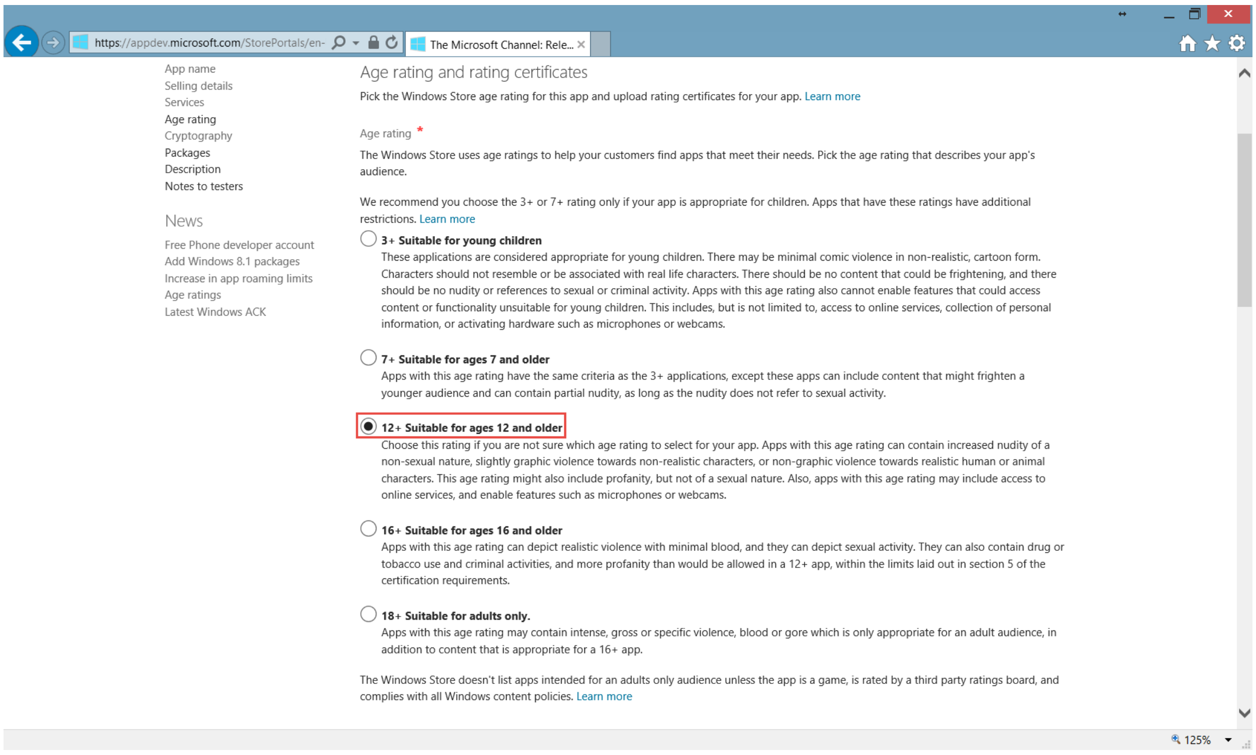
Product ID	Price tier	Product lifetime	Content type
	Pick a price tier	Forever	Inherit from app

Add another offer

Save

Age Rating

Selecting “12+” in the age rating is a good choice for “normal” apps that aren’t specifically targeted for children. If you select a lower age rating, the app will be tested against different criteria to make sure it’s suitable for younger children. If your app contains graphic violence however, you will want to set the age rating higher.


https://appdev.microsoft.com/StorePortals/en-
The Microsoft Channel: Rele...
125%

App name
Selling details
Services
Age rating
Cryptography
Packages
Description
Notes to testers

News

Free Phone developer account
Add Windows 8.1 packages
Increase in app roaming limits
Age ratings
Latest Windows ACK

Age rating and rating certificates

Pick the Windows Store age rating for this app and upload rating certificates for your app. Learn more

Age rating *

The Windows Store uses age ratings to help your customers find apps that meet their needs. Pick the age rating that describes your app's audience.

We recommend you choose the 3+ or 7+ rating only if your app is appropriate for children. Apps that have these ratings have additional restrictions. Learn more

○ **3+ Suitable for young children**
These applications are considered appropriate for young children. There may be minimal comic violence in non-realistic, cartoon form. Characters should not resemble or be associated with real life characters. There should be no content that could be frightening, and there should be no nudity or references to sexual or criminal activity. Apps with this age rating also cannot enable features that could access content or functionality unsuitable for young children. This includes, but is not limited to, access to online services, collection of personal information, or activating hardware such as microphones or webcams.

○ **7+ Suitable for ages 7 and older**
Apps with this age rating have the same criteria as the 3+ applications, except these apps can include content that might frighten a younger audience and can contain partial nudity, as long as the nudity does not refer to sexual activity.

[✓ **12+ Suitable for ages 12 and older**]
Choose this rating if you are not sure which age rating to select for your app. Apps with this age rating can contain increased nudity of a non-sexual nature, slightly graphic violence towards non-realistic characters, or non-graphic violence towards realistic human or animal characters. This age rating might also include profanity, but not of a sexual nature. Also, apps with this age rating may include access to online services, and enable features such as microphones or webcams.

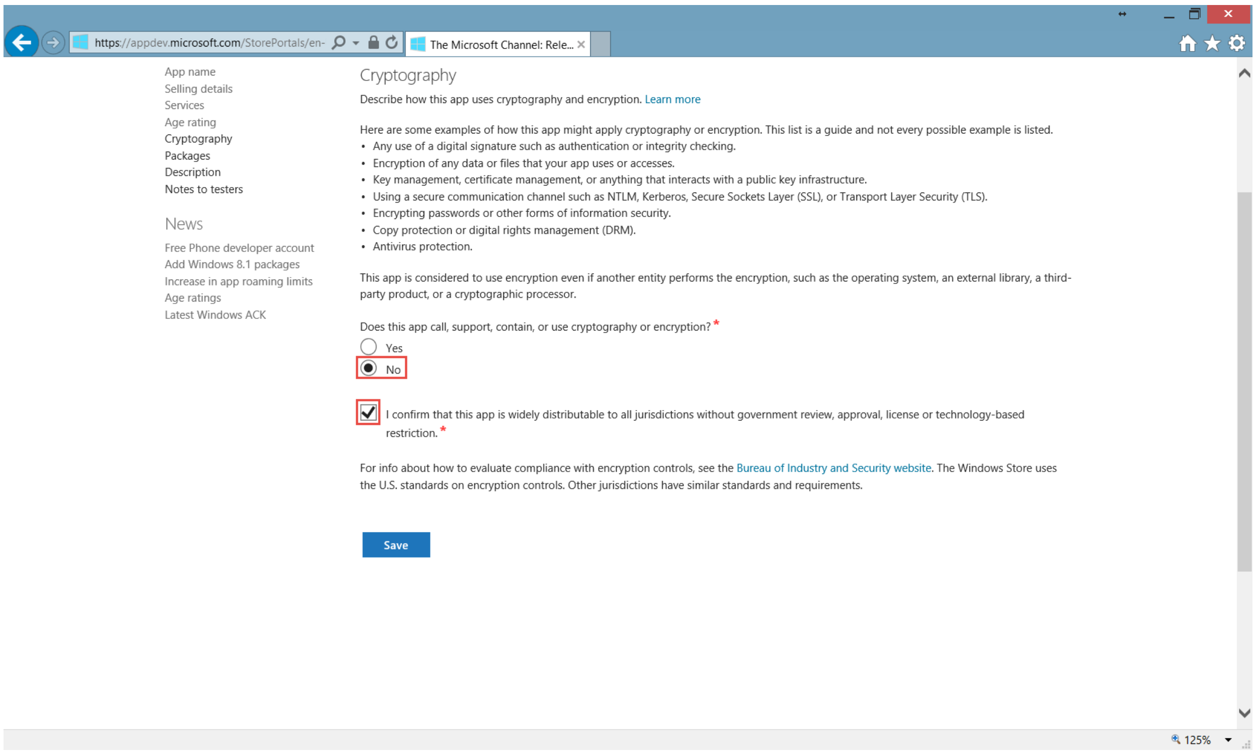
○ **16+ Suitable for ages 16 and older**
Apps with this age rating can depict realistic violence with minimal blood, and they can depict sexual activity. They can also contain drug or tobacco use and criminal activities, and more profanity than would be allowed in a 12+ app, within the limits laid out in section 5 of the certification requirements.

○ **18+ Suitable for adults only.**
Apps with this age rating may contain intense, gross or specific violence, blood or gore which is only appropriate for an adult audience, in addition to content that is appropriate for a 16+ app.

The Windows Store doesn't list apps intended for an adults only audience unless the app is a game, is rated by a third party ratings board, and complies with all Windows content policies. Learn more

Cryptography

As our app doesn't make use of any cryptographic libraries or functions, answer the question with "No" and confirm by selecting the checkbox.


https://appdev.microsoft.com/StorePortals/en-
The Microsoft Channel: Rele...
Save
125%

App name
Selling details
Services
Age rating
Cryptography
Packages
Description
Notes to testers

News

Free Phone developer account
Add Windows 8.1 packages
Increase in app roaming limits
Age ratings
Latest Windows ACK

Cryptography

Describe how this app uses cryptography and encryption. Learn more

Here are some examples of how this app might apply cryptography or encryption. This list is a guide and not every possible example is listed.

- Any use of a digital signature such as authentication or integrity checking.
- Encryption of any data or files that your app uses or accesses.
- Key management, certificate management, or anything that interacts with a public key infrastructure.
- Using a secure communication channel such as NTLM, Kerberos, Secure Sockets Layer (SSL), or Transport Layer Security (TLS).
- Encrypting passwords or other forms of information security.
- Copy protection or digital rights management (DRM).
- Antivirus protection.

This app is considered to use encryption even if another entity performs the encryption, such as the operating system, an external library, a third-party product, or a cryptographic processor.

Does this app call, support, contain, or use cryptography or encryption? *

Yes [✓ No]

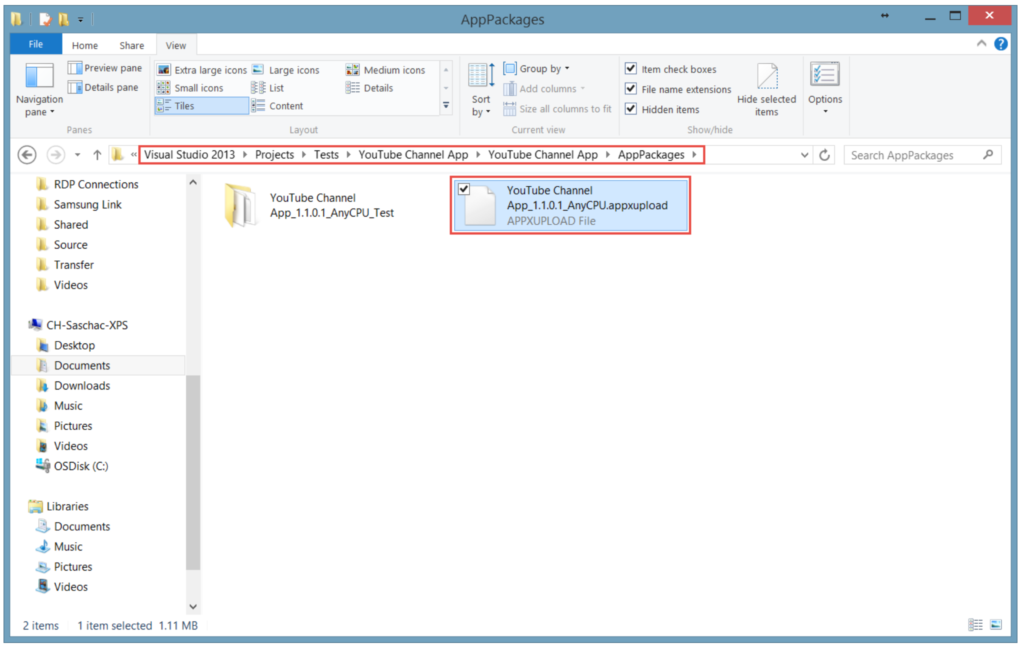
✓ I confirm that this app is widely distributable to all jurisdictions without government review, approval, license or technology-based restriction. *

For info about how to evaluate compliance with encryption controls, see the Bureau of Industry and Security website. The Windows Store uses the U.S. standards on encryption controls. Other jurisdictions have similar standards and requirements.

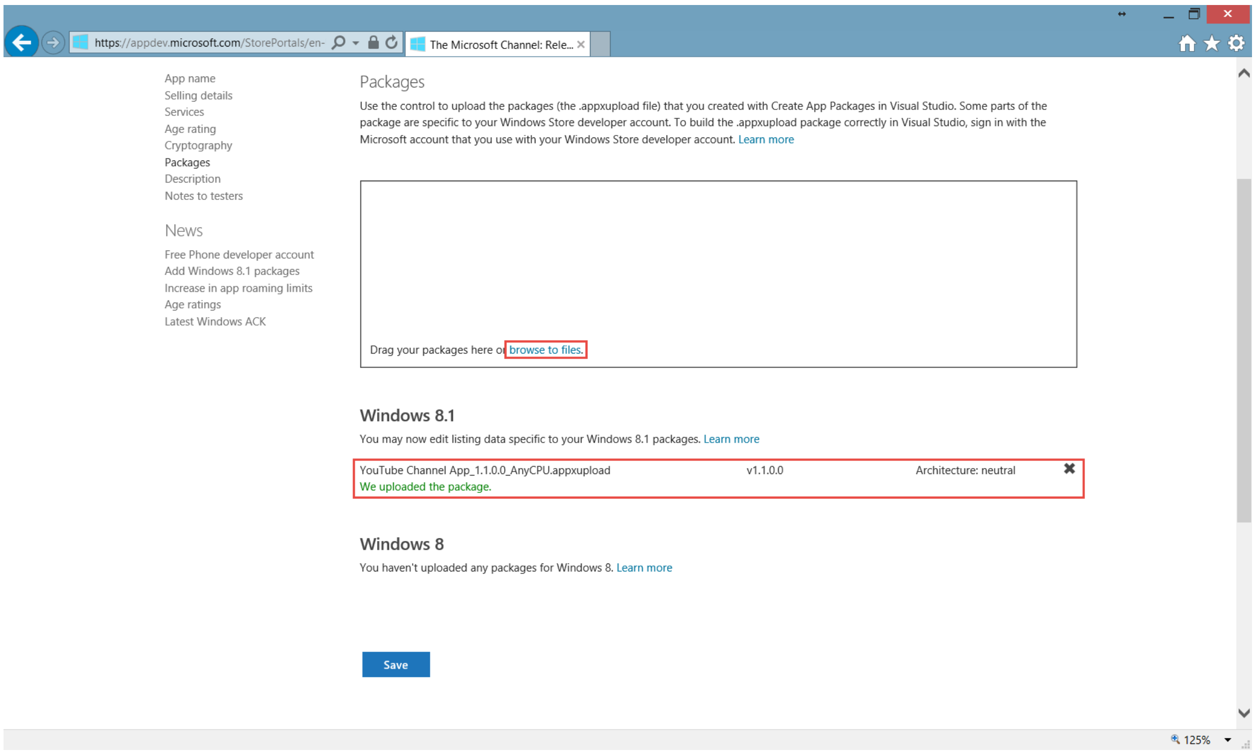
Packages

In here we upload our app package generated by Visual Studio. Click the "browse to files" link and point to the folder "AppPackages" inside your Visual Studio solution. This folder has been generated when we created the package and you will find a .appxupload file for each version/app package we have created.

Pick the latest .appxupload file. You can ignore the folder with the same name as the package, it is only used if you want to share your app with someone for testing it.


AppPackages
File Home Share View
Visual Studio 2013 › Projects › Tests › YouTube Channel App › YouTube Channel App › AppPackages
YouTube Channel App_1.1.0.1_AnyCPU_Test
YouTube Channel App_1.1.0.1_AnyCPU.appxupload
APPXUPLOAD File
2 items 1 item selected 1.11 MB

After the upload completes, you will see the package appear on the Dev Center under the “Windows 8.1” title. We didn’t provide a Windows 8, only a Windows 8.1 version, so this is correct.



Packages

Use the control to upload the packages (the .appxupload file) that you created with Create App Packages in Visual Studio. Some parts of the package are specific to your Windows Store developer account. To build the .appxupload package correctly in Visual Studio, sign in with the Microsoft account that you use with your Windows Store developer account. Learn more

Drag your packages here or browse to files.

Windows 8.1

You may now edit listing data specific to your Windows 8.1 packages. Learn more

YouTube Channel App_1.1.0.0_AnyCPU.appxupload v1.1.0.0 Architecture: neutral
We uploaded the package.

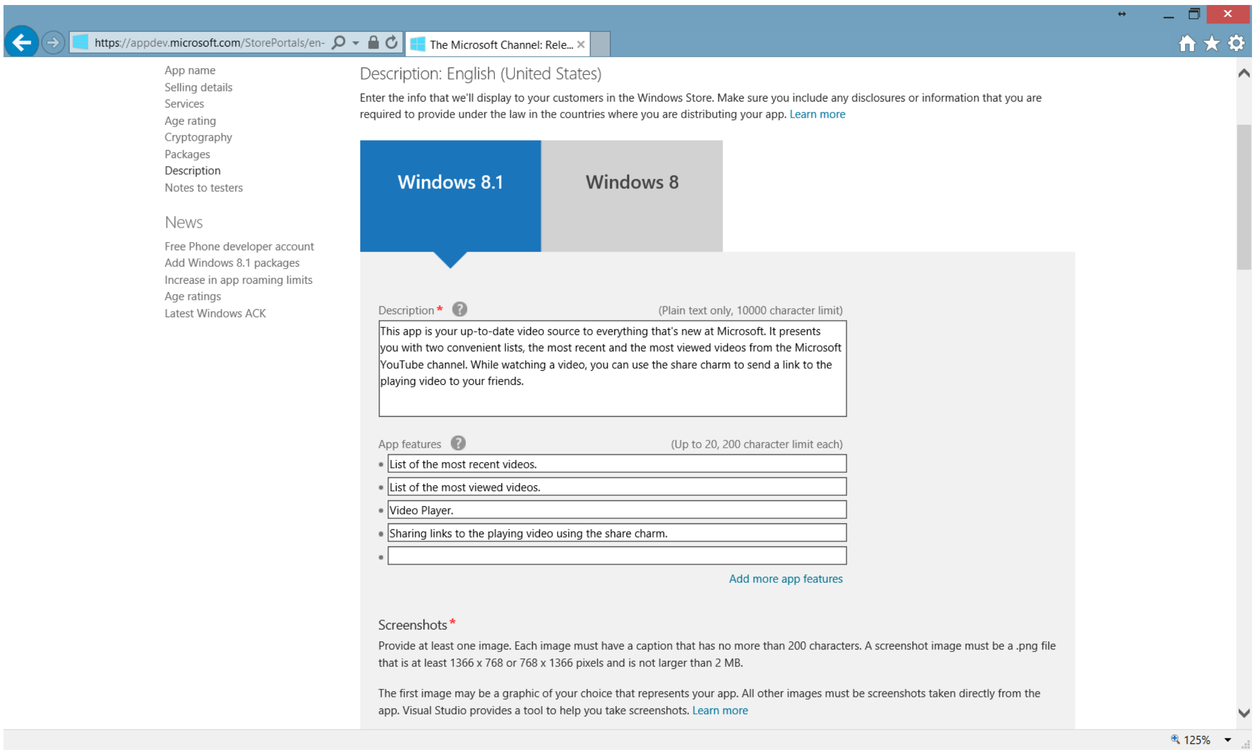
Windows 8

You haven't uploaded any packages for Windows 8. Learn more

Description

Here, you will find a description page for every version (Windows 8, Windows 8.1) and every language you have created the app in. We will only have to provide one set of data as our app is English only and runs on Windows 8.1.

Add a description and a short list of app features. Feel free to steal from the text below but make sure to use your own YouTube channel name.



Description: English (United States)

Enter the info that we'll display to your customers in the Windows Store. Make sure you include any disclosures or information that you are required to provide under the law in the countries where you are distributing your app. Learn more

Description * (Plain text only, 10000 character limit)

This app is your up-to-date video source to everything that's new at Microsoft. It presents you with two convenient lists, the most recent and the most viewed videos from the Microsoft YouTube channel. While watching a video, you can use the share charm to send a link to the playing video to your friends.

App features (Up to 20, 200 character limit each)

- List of the most recent videos.
- List of the most viewed videos.
- Video Player.
- Sharing links to the playing video using the share charm.
-

Add more app features

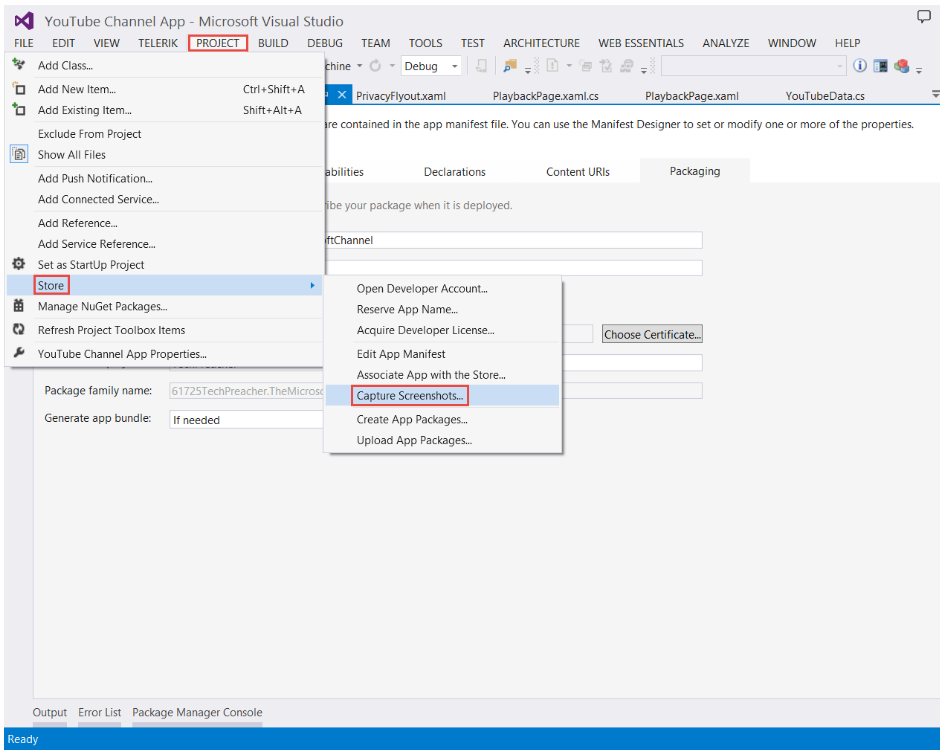
Screenshots *

Provide at least one image. Each image must have a caption that has no more than 200 characters. A screenshot image must be a .png file that is at least 1366 x 768 or 768 x 1366 pixels and is not larger than 2 MB.

The first image may be a graphic of your choice that represents your app. All other images must be screenshots taken directly from the app. Visual Studio provides a tool to help you take screenshots. Learn more

Description – Screenshots

The quickest way to get to the screen shots in the format required by the store (1366px * 768px, <2MB) is to head back to Visual Studio and select “Project”, “Store” and “Capture Screenshots” from the pulldown menus.



re contained in the app manifest file. You can use the Manifest Designer to set or modify one or more of the properties.

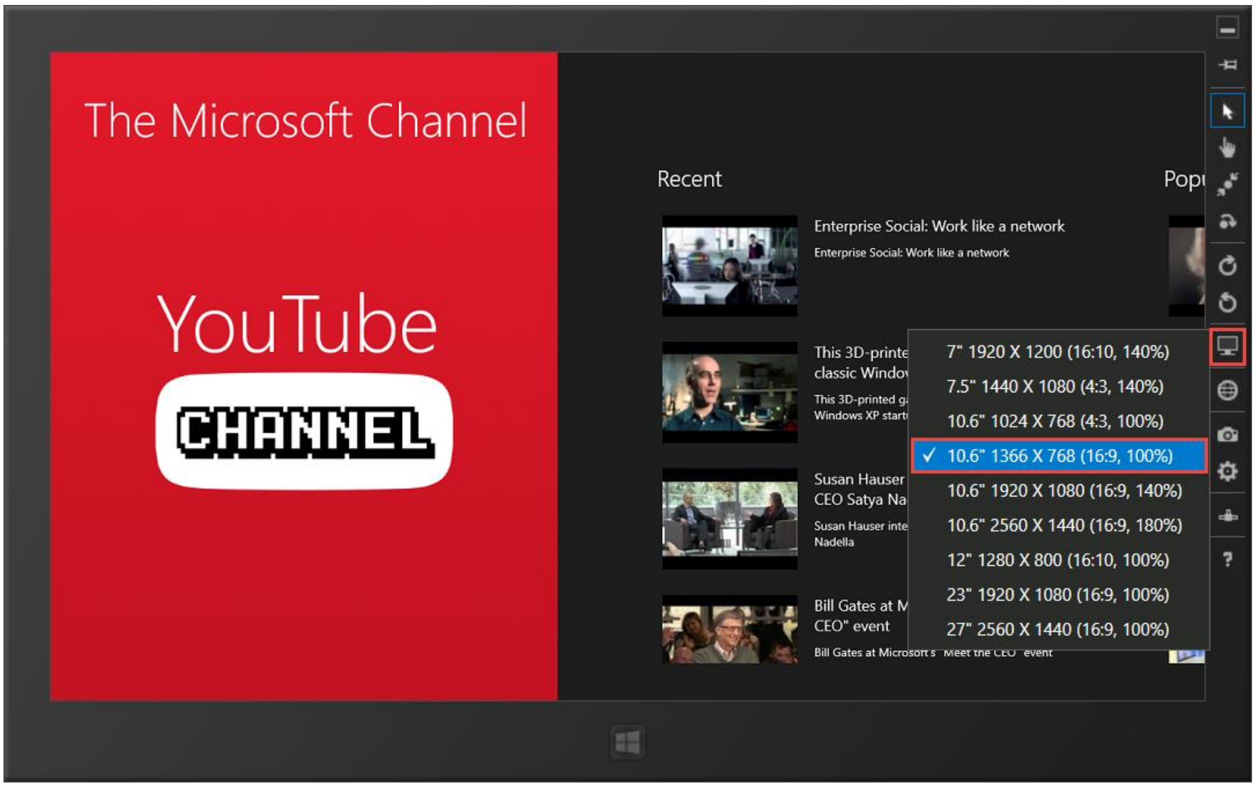
ibe your package when it is deployed.

ftChannel

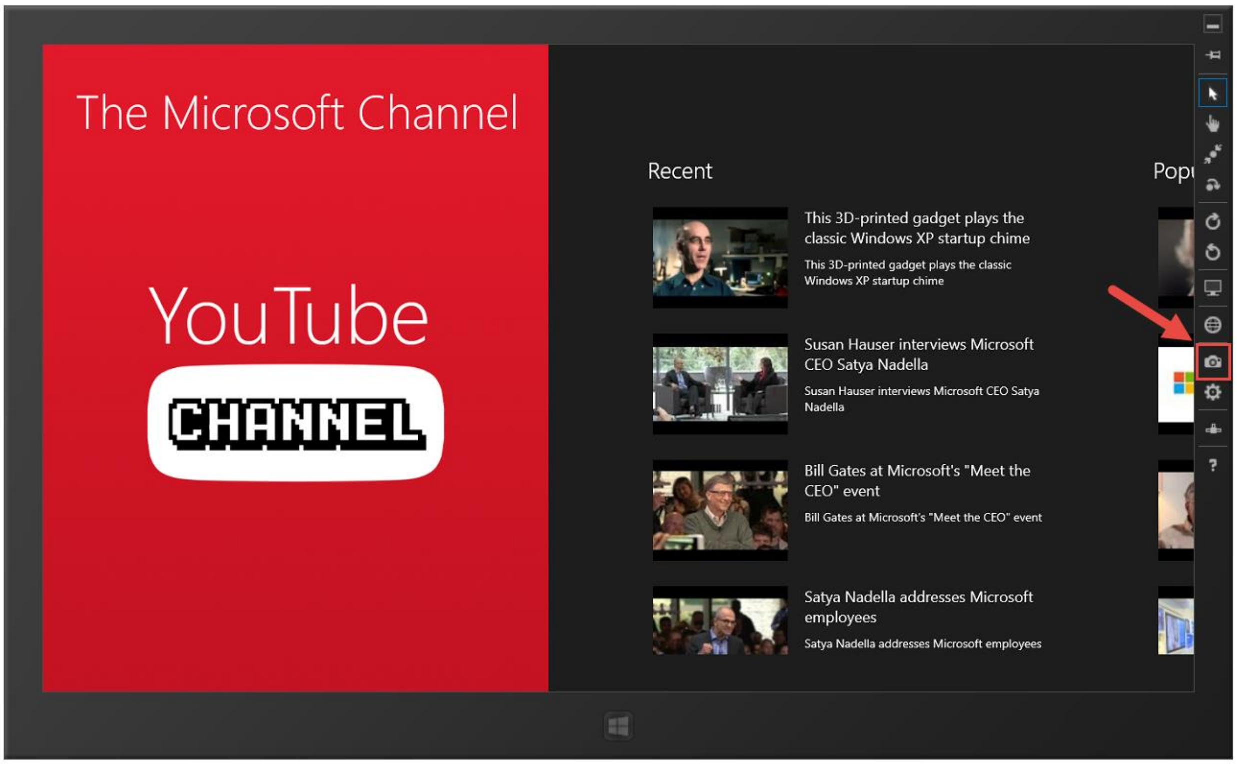
Package family name: 61725TechPreacher.TheMicros

Generate app bundle: If needed

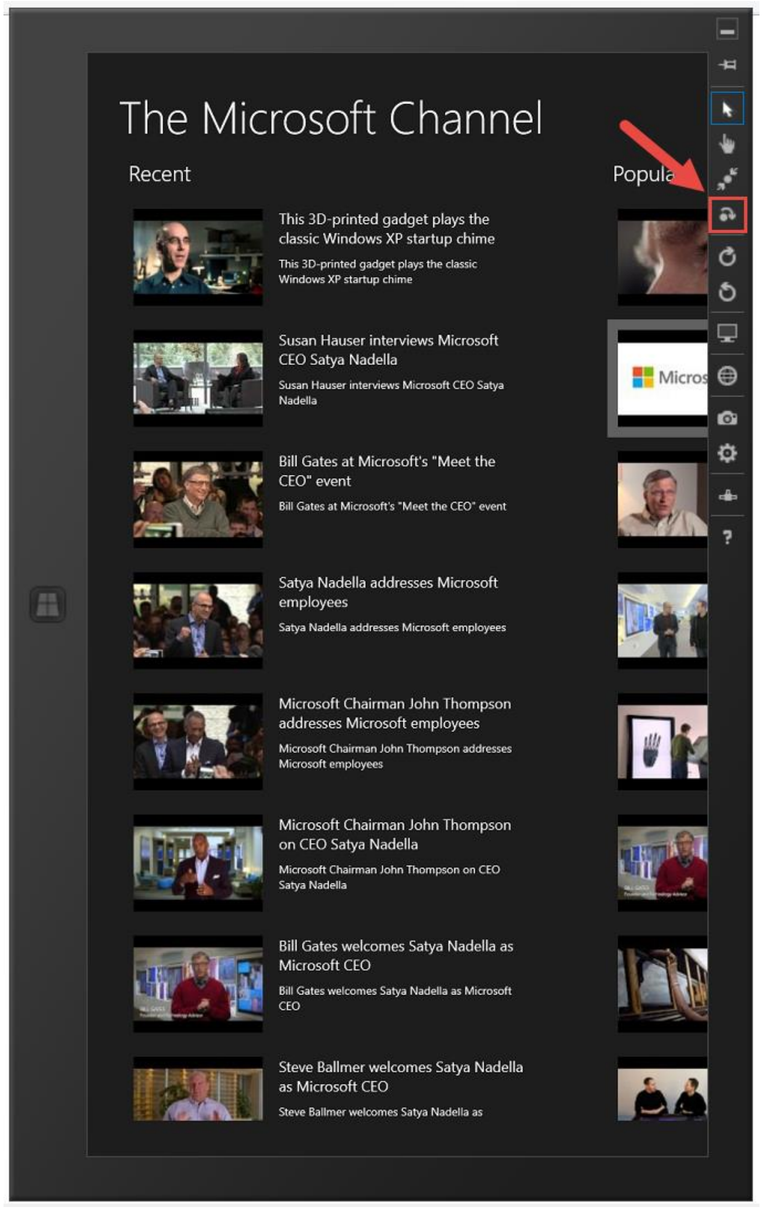
This will start the Simulator running your app. Make sure you have the right resolution set.


The Microsoft Channel
YouTube
CHANNEL
Recent
Popu
Enterprise Social: Work like a network
Enterprise Social: Work like a network
This 3D-printe
classic Windo
Susan Hauser
CEO Satya Na
Bill Gates at M
CEO" event
7" 1920 X 1200 (16:10, 140%)
7.5" 1440 X 1080 (4:3, 140%)
10.6" 1024 X 768 (4:3, 100%)
✓ 10.6" 1366 X 768 (16:9, 100%)
10.6" 1920 X 1080 (16:9, 140%)
10.6" 2560 X 1440 (16:9, 180%)
12" 1280 X 800 (16:10, 100%)
23" 1920 X 1080 (16:9, 100%)
27" 2560 X 1440 (16:9, 100%)

Now navigate to the different views in your app (the main page, the main page scrolled to the right, the video player and maybe the sharing charm in action) and press the “Screenshot” button every time you like the look.


The Microsoft Channel
YouTube
CHANNEL
Recent
Popu
This 3D-printed gadget plays the classic Windows XP startup chime
This 3D-printed gadget plays the classic Windows XP startup chime
Susan Hauser interviews Microsoft CEO Satya Nadella
Susan Hauser interviews Microsoft CEO Satya Nadella
Bill Gates at Microsoft's "Meet the CEO" event
Bill Gates at Microsoft's "Meet the CEO" event
Satya Nadella addresses Microsoft employees
Satya Nadella addresses Microsoft employees

You can also rotate your simulator to test and screenshot your app running in a vertical resolution by pressing the rotate button.



The Microsoft Channel

Recent

This 3D-printed gadget plays the classic Windows XP startup chime

This 3D-printed gadget plays the classic Windows XP startup chime

Susan Hauser interviews Microsoft CEO Satya Nadella

Susan Hauser interviews Microsoft CEO Satya Nadella

Bill Gates at Microsoft's "Meet the CEO" event

Bill Gates at Microsoft's "Meet the CEO" event

Satya Nadella addresses Microsoft employees

Satya Nadella addresses Microsoft employees

Microsoft Chairman John Thompson addresses Microsoft employees

Microsoft Chairman John Thompson addresses Microsoft employees

Microsoft Chairman John Thompson on CEO Satya Nadella

Microsoft Chairman John Thompson on CEO Satya Nadella

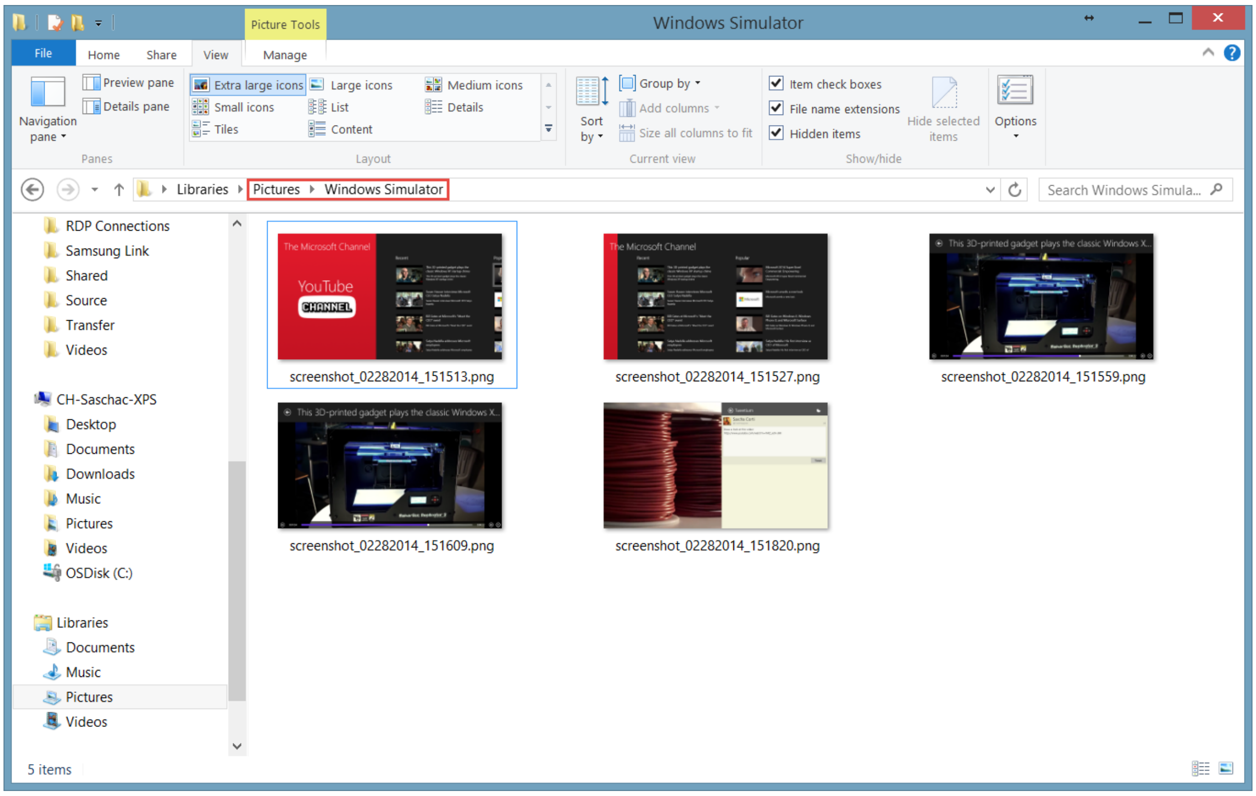
Bill Gates welcomes Satya Nadella as Microsoft CEO

Bill Gates welcomes Satya Nadella as Microsoft CEO

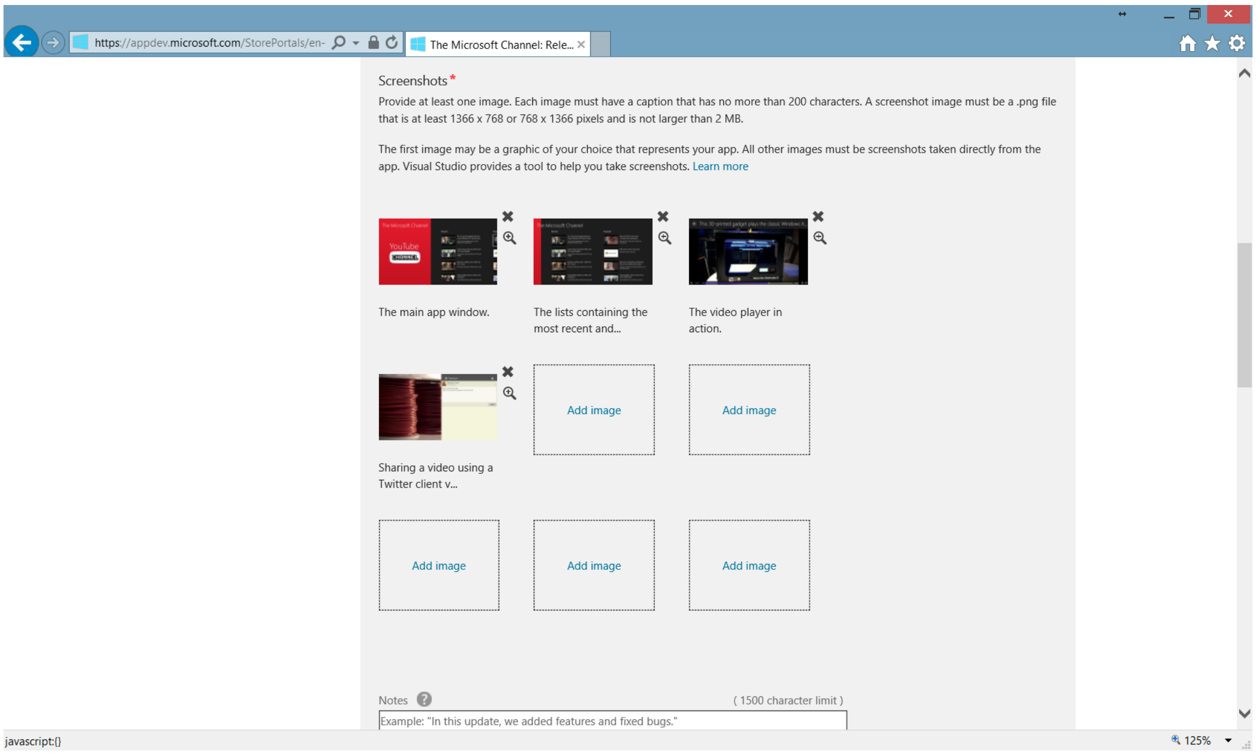
Steve Ballmer welcomes Satya Nadella as Microsoft CEO

Steve Ballmer welcomes Satya Nadella as

All the screenshots are automatically saved in the “My Pictures\Windows Simulator” folder. Make sure you have all the screenshots you need.



Head over to the Dev Center and click on the “Add image” links to upload all your screenshots. Don’t forget to add a short description of what can be seen in the picture.



Screenshots*

Provide at least one image. Each image must have a caption that has no more than 200 characters. A screenshot image must be a .png file that is at least 1366 x 768 or 768 x 1366 pixels and is not larger than 2 MB.

The first image may be a graphic of your choice that represents your app. All other images must be screenshots taken directly from the app. Visual Studio provides a tool to help you take screenshots. Learn more

The main app window.

The lists containing the most recent and...

The video player in action.

Sharing a video using a Twitter client v...

Add image

Add image

Add image

Add image

Add image

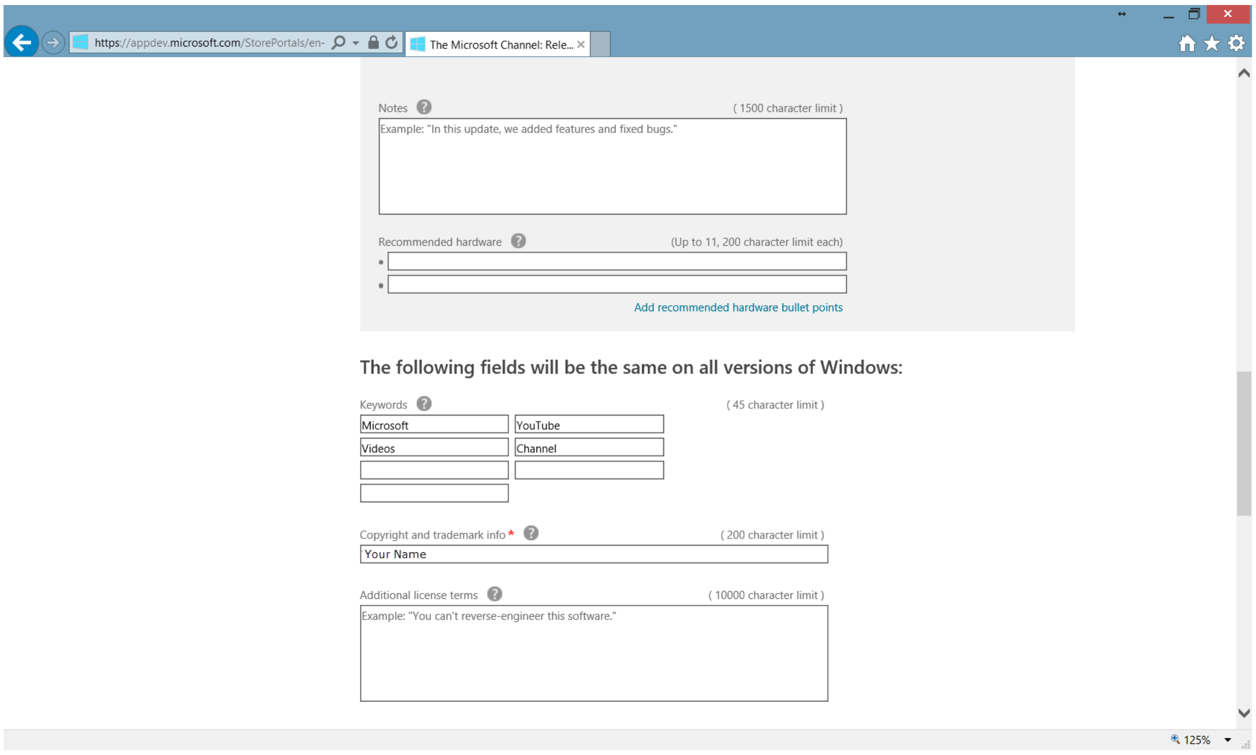
Notes (1500 character limit)

Example: "In this update, we added features and fixed bugs."

https://appdev.microsoft.com/StorePortals/en-
The Microsoft Channel: Rele...
javascript:{}
125%

Description – Keywords

Add some keywords relevant to your app and your Dev Center publisher name in the “Copyright” field.



https://appdev.microsoft.com/StorePortals/en- | The Microsoft Channel: Rele...

Notes (1500 character limit)
Example: "In this update, we added features and fixed bugs."

Recommended hardware (Up to 11, 200 character limit each)

Add recommended hardware bullet points

The following fields will be the same on all versions of Windows:

Keywords (45 character limit)

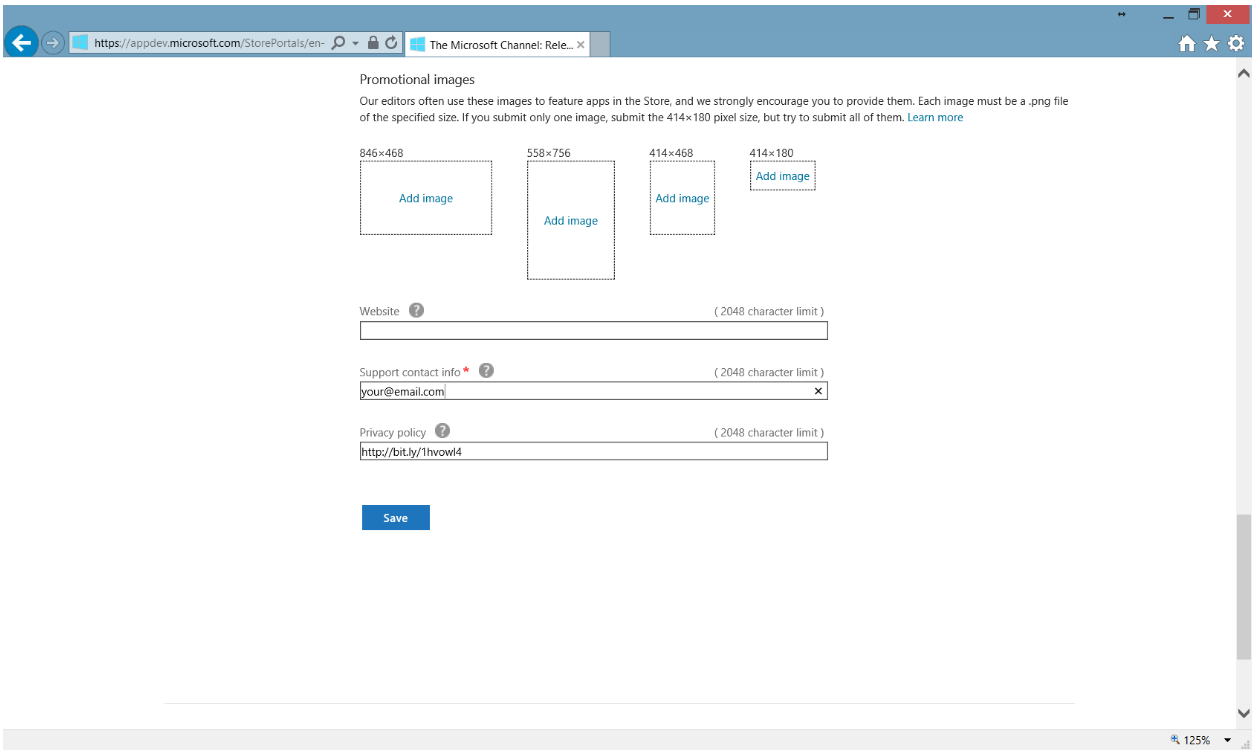
Microsoft	YouTube
Videos	Channel

Copyright and trademark info * (200 character limit)
Your Name

Additional license terms (10000 character limit)
Example: "You can't reverse-engineer this software."

In the "Support Contact Info" field, type the email address you have used when creating your privacy policy.

Get the link to your online privacy policy and shorten it using any URL shortener. I have used https://bitly.com/. Paste the shortened link into the "Privacy Policy" field and make sure, it really works! Your app would fail certification if this link was wrong and that would be a pity!



https://appdev.microsoft.com/StorePortals/en- | The Microsoft Channel: Rele...

Promotional images
Our editors often use these images to feature apps in the Store, and we strongly encourage you to provide them. Each image must be a .png file of the specified size. If you submit only one image, submit the 414×180 pixel size, but try to submit all of them. Learn more

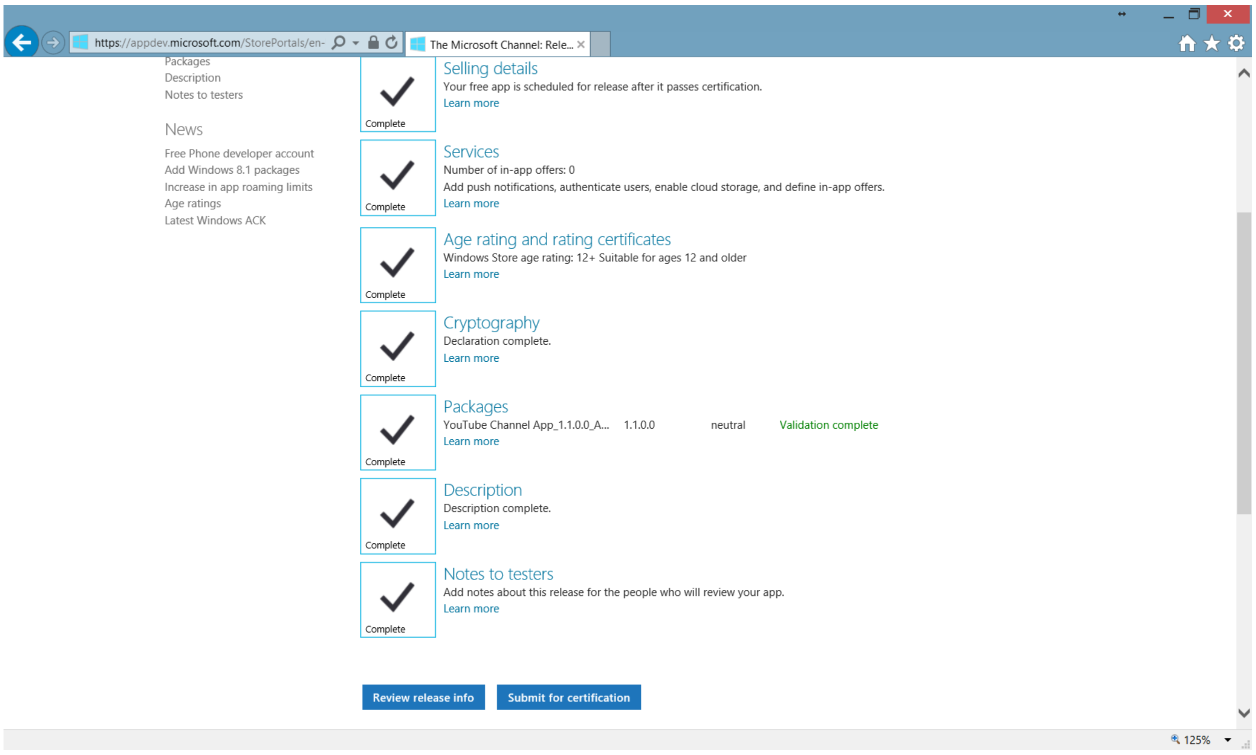
Website (2048 character limit)

Support contact info * (2048 character limit)
your@email.com

Privacy policy (2048 character limit)
http://bit.ly/1hvowl4

Notes to testers

There are no special testing requirements for our app, so this section can be left blank.



https://appdev.microsoft.com/StorePortals/en-

The Microsoft Channel: Rele...

Packages
Description
Notes to testers

News
Free Phone developer account
Add Windows 8.1 packages
Increase in app roaming limits
Age ratings
Latest Windows ACK

Complete — Selling details
Your free app is scheduled for release after it passes certification.
Learn more

Complete — Services
Number of in-app offers: 0
Add push notifications, authenticate users, enable cloud storage, and define in-app offers.
Learn more

Complete — Age rating and rating certificates
Windows Store age rating: 12+ Suitable for ages 12 and older
Learn more

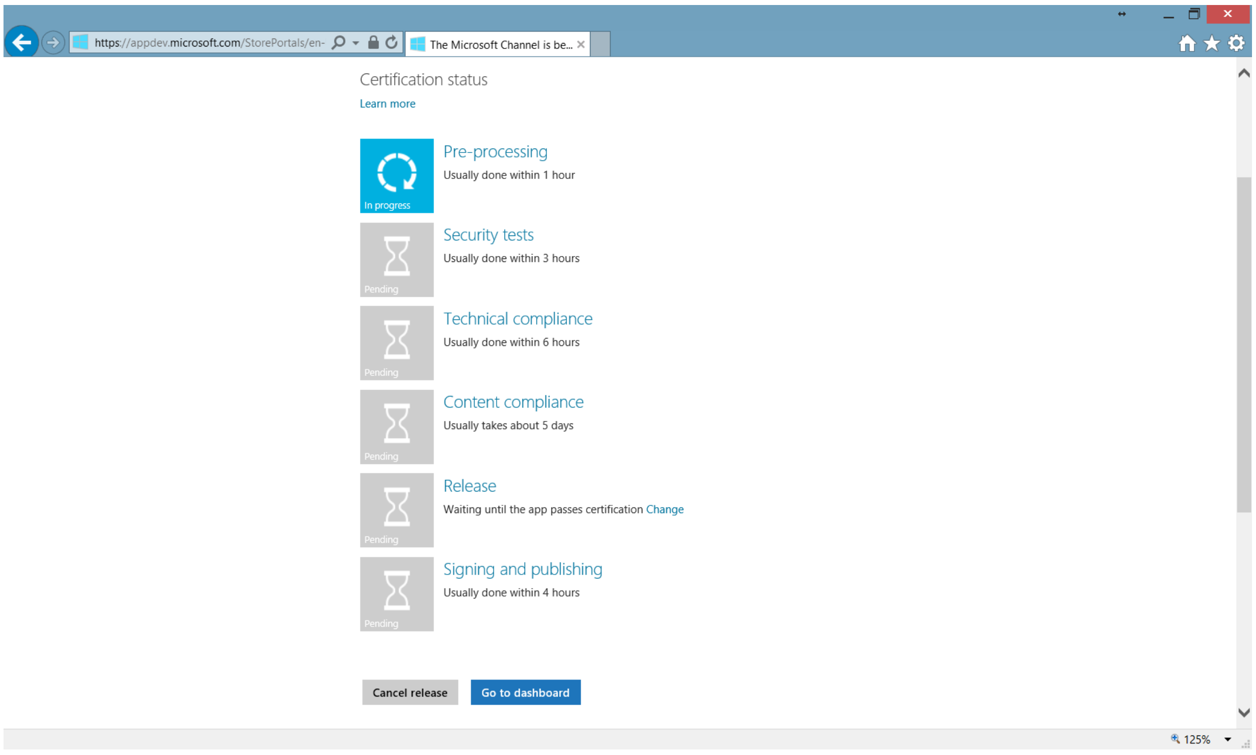
Complete — Cryptography
Declaration complete.
Learn more

Complete — Packages
YouTube Channel App_1.1.0.0_A... 1.1.0.0 neutral Validation complete
Learn more

Complete — Description
Description complete.
Learn more

Complete — Notes to testers
Add notes about this release for the people who will review your app.
Learn more

Once all the steps show as “Complete”, you can submit your app for publishing to the Windows Store by pressing “Submit for Certification”. You will see a summary of the certification progress.



https://appdev.microsoft.com/StorePortals/en-

The Microsoft Channel is be...

Certification status
Learn more

In progress — Pre-processing
Usually done within 1 hour

Pending — Security tests
Usually done within 3 hours

Pending — Technical compliance
Usually done within 6 hours

Pending — Content compliance
Usually takes about 5 days

Pending — Release
Waiting until the app passes certification Change

Pending — Signing and publishing
Usually done within 4 hours

Congratulations! Your app should become available in the Windows Store within a few hours to a few days.

This concludes the hands-on lab.