



**Vendor: Microsoft**

**Exam Code: 70-485**

**Exam Name: Advanced Windows Store App Development**

**using C#**

**Version: Demo**

## Topic 1, Scenario Margie's Travel

### Background

You are developing a Windows Store media sharing app for the sales and marketing team at Margie's Travel. The app will allow team members to download documents and media about current and proposed products and services from the company's cloud-based media manager service. Team members will be able to add new content to the cloud service and to print and share content.

### Business Requirements

#### Behavior

Team members must be able to download product information data sheets, marketing materials, and product demonstration video clips from the company's server.

Team members must be able to select and upload multiple files that contain new and modified content as a batch.

Team members must be able to stream video clips to other devices in the vicinity of the team member's device. The app will not support the streaming of photographs.

The app must allow team members to pause, restart, or cancel uploads and downloads of files.

The app must report both the progress and completion status of these operations. It must also return results about upload and download operations.

### User Interface

The app must include a photo viewer. When photos are added or deleted in the photo viewer window, they must animate in and out of the field of view. Remaining photos must move to fill the empty space created when photos are deleted. The photo viewer must support semantic zoom.

The app must display information on the lock screen of the device. The information must include text-based alerts and a value indicating the number of pending file downloads.

### Technical Requirements

#### Behavior

The company has an existing component named VideoProcessor. This component compresses video clips and performs other processing before the video clips are uploaded to the media manager service. The component was written with managed code. The VideoProcessor component will also be used by Windows Store apps developed in HTML5 and JavaScript. The apps must be able to call the overload of the ProcessVideo() method that accepts a string and a Boolean value as parameters.

When a team member selects a video clip to download, the app must download the file as a background task. After a download has started, the app should maintain the network connection to the server even when the app is suspended.

### User Interface

The app must include a custom photo viewer control. The control will be updated frequently and may be deployed separately from the rest of the app. The photo viewer control must support templates and styles.

### User Interface

The app must include a custom photo viewer control. The control will be updated frequently and may be deployed separately from the rest of the app. The photo viewer control must support templates and styles.

The app must use a Grid control as the root layout control. The photo viewer must be placed in the second row of the grid.

The appearance of the app must change when the app is in snapped mode. The first row of the root layout grid must not change height. The second row must fill all available space.

Available video clips must be displayed in an extended ListView control class named DownloadedVideoList.

The template for the DownloadedVideoList is already defined.

New video clips should be added to DownloadedVideoList when the DownloadVideoQ method completes.

New video clip items in the DownloadedVideoList should color change periodically to alert the team member.

### Application Structure

Relevant portions of the app files are as follows. (Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.)

#### App.xaml.cs

```

AP01 cts= newCancellationTokenSource();
AP02 private List<DownloadOperation>MyPendingDownloads;
AP03 privateasyncTaskHandleMyPendingDownloads(DownloadOperationdownload,
boolstart)
AP04 {
AP05     MyPendingDownloads.Add(download);
AP06     Progress<DownloadOperation> progressCallback = new
Progress<DownloadOperation>(DownloadProgress);
AP07     if(start)
AP08     {
AP09         awaitdownload.StartAsync().AsTask(cts.Token, progressCallback);
AP10     }
AP11     else
AP12     {
AP13         awaitdownload.AttachAsync().AsTask(cts.Token, progressCallback);
AP14     }
AP15 }
AP16 privateasyncvoidUploadContent()
AP17 {
AP18     FileOpenPickerpicker = newFileOpenPicker();
AP19
AP20     List<BackgroundTransferContentPart> uploadGrp = new
List<BackgroundTransferContentPart>();
AP21     for(intfileNum = 0; fileNum < files.Count; fileNum ++)
AP22     {
AP23         BackgroundTransferContentPartuploadItem= new
BackgroundTransferContentPart("File"+ fileNum,
files[fileNum].Name);
AP24         uploadItem.SetFile(files[fileNum]);
AP25         uploadGrp.Add(uploadItem);
AP26     }
AP27     BackgroundUploaderuploader = newBackgroundUploader();
AP28
AP29     awaitHandleUploadAsync(upload, true);
AP30 }

```

### VideoProcessor.cs

```
IP01 publicclassVideoProcessor
IP02 {
IP03
IP04     publicVideoProcessor(intvideoID)
IP05     {
IP06         ...
IP07     }
IP08
IP09     publicVideoProcessor(stringvideoName)
IP10     {
IP11         ...
IP12     }
IP13
IP14
IP15     publicvoidProcessVideo(stringvideoName, stringvideoType)
IP16     {
IP17         ...
IP18     }
IP19
IP20     publicvoidProcessVideo(stringvideoName, boolcompressFile)
IP21     {
IP22         ...
IP23     }
IP24 }
```

### MainPage.xaml

```
MP01 <Grid x:Name="LayoutRoot">
MP02     <Grid.RowDefinitions>
MP03         <RowDefinitionHeight="100"/>
MP04         <RowDefinitionHeight="200"/>
MP05     </Grid.RowDefinitions>
MP06     <VisualStateManager.VisualStateGroups>
MP07
MP08     </VisualStateManager.VisualStateGroups>
MP09 </Grid>
```

### MainPage.xaml.cs

```
MC01 private PlayToManagerptMgr = PlayToManager.GetForCurrentView();
MC02
MC03 protectedoverridevoidOnNavigatedTo(NavigationEventArgs)
MC04 {
MC05
MC06
MC07 }
MC08 privatevoidSourceRequestHandler(PlayToManagersender,
    PlayToSourceRequestedEventArgs)
MC09 {
MC10
MC11     e.SourceRequest.SetSource(mediaElement.PlayToSource);
MC12 }
MC13 publicvoidStartNewVideoAnimation()
MC14 {
MC15     NewVideoStoryboard.Begin();
MC16 }
MC17 publicvoidDownloadVideo(stringvideoName)
MC18 {
MC19     ...
MC20     videoList.Items.Add(videoName);
MC21 }
```

**QUESTION 1**

**HOTSPOT**

You need to meet the business requirements about downloading and uploading.

How should you configure the app?

To answer, select the appropriate options from each drop-down list in the answer area.

Configure the Application UI settings in Package.appxmanifest

Lock screen notifications:

  
Tile  
Badge and Tile Text  
Wide Logo Only

Logo files:

  
Tile Image Only  
Badge Logo and Tile Image  
Badge Logo and Wide Logo

Configure the Declarations settings in Package.appxmanifest

Add a Background Task declaration and configure support for the following task types:

  
Photo file stream  
Control channel  
User actions  
Device availability  
Playback status

**Correct Answer:**

Configure the Application UI settings in Package.appxmanifest

Lock screen notifications:

▼

- Tile
- Badge and Tile Text
- Wide Logo Only

Logo files:

▼

- Tile Image Only
- Badge Logo and Tile Image
- Badge Logo and Wide Logo

Configure the Declarations settings in Package.appxmanifest

Add a Background Task declaration and configure support for the following task types:

▼

- Photo file stream
- Control channel
- User actions
- Device availability
- Playback status

**QUESTION 2**

You need to implement the business requirement to display video clips. Which code segment should you use in the MainPage.xaml file?

- A. 

```
<DownloadedVideoList x:Name="videoList">
  <DownloadedVideoList.Resources>
    <Storyboard x:Name="NewVideoStoryboard">
      <ColorAnimation Storyboard.TargetName="NewVideoBrush"
        Storyboard.TargetProperty="Color" From="Red" To="Green"
        Duration="0:0:8" RepeatBehavior="Forever"/>
    </Storyboard>
  </DownloadedVideoList.Resources>
  <DownloadedVideoList.Background>
    <SolidColorBrush x:Name="NewVideoBrush" Color="Red"/>
  </DownloadedVideoList.Background>
</DownloadedVideoList>
```
- B. 

```
<DownloadedVideoList x:Name="videoList">
  <DownloadedVideoList.Resources>
    <Storyboard x:Name="NewVideoStoryboard">
      <ColorAnimation Storyboard.TargetName="NewVideoBrush"
        Storyboard.TargetProperty="Color" From="Red" To="Green"
        AutoReverse="true"/>
    </Storyboard>
  </DownloadedVideoList.Resources>
  <DownloadedVideoList.Background>
    <SolidColorBrush x:Name="NewVideoBrush" Color="Red"/>
  </DownloadedVideoList.Background>
</DownloadedVideoList>
```
- C. 

```
<DownloadedVideoList x:Name="videoList">
  <DownloadedVideoList.Transitions>
    <TransitionCollection>
      <EntranceThemeTransition/>
    </TransitionCollection>
  </DownloadedVideoList.Transitions>
</DownloadedVideoList>
```
- D. 

```
<DownloadedVideoList x:Name="videoList">
  <DownloadedVideoList.Transitions>
    <TransitionCollection>
      <AddDeleteThemeTransition/>
    </TransitionCollection>
  </DownloadedVideoList.Transitions>
</DownloadedVideoList>
```

- A. Option A  
B. Option B  
C. Option C  
D. Option D

**Correct Answer: A**



**QUESTION 3**

You need to implement downloading of media files and other content. Which code segment should you add to App.xaml.cs?

- A. 

```
private async Task GetPendingDownloadsList()
{
    IReadOnlyList<DownloadOperation> downloads = await
        BackgroundDownloader.GetCurrentDownloadsAsync();
    if (downloads.Count > 0)
    {
        List<Task> myTasks = new List<Task>();
        for (int i=0; i < downloads.count; i++)
        {
            await HandleMyPendingDownloads(downloads[i], true);
        }
        await Task.WhenAll(myTasks);
    }
}
```
- B. 

```
private async Task GetPendingDownloadsList()
{
    IReadOnlyList<DownloadOperation> downloads = await
        BackgroundDownloader.GetCurrentDownloadsAsync();
    if (downloads.Count > 0)
    {
        List<Task> myTasks = new List<Task>();
        foreach (DownloadOperation download in downloads)
        {
            myTasks.Add(HandleDownloadAsync(download, false));
        }
        await Task.WhenAll(myTasks);
    }
}
```
- C. 

```
private GetPendingDownloadsList()
{
    IReadOnlyList<DownloadOperation> downloads = await
        BackgroundDownloader.GetCurrentDownloadsAsync();
    if (downloads.Count > 0)
    {
        List<Task> myTasks = new List<Task>();
        for (int i=0; i < downloads.count; i++)
        {
            await HandleMyPendingDownloads(downloads[i], true);
        }
        await Task.WhenAll(myTasks);
    }
}
```
- D. 

```
private Task GetPendingDownloadsList()
{
    IReadOnlyList<DownloadOperation> downloads =
        BackgroundDownloader.CreateDownloadAsync();
    if (downloads.Count > 0)
    {
        List<Task> myTasks = new List<Task>();
        foreach (DownloadOperation download in downloads)
        {
            myTasks.Add(HandleDownloadAsync(download, false));
        }
        Task.WhenAll(myTasks);
    }
}
```

- A. Option A
- B. Option B
- C. Option C
- D. Option D

**Correct Answer:** B

#### QUESTION 4

You need to ensure that the app uploads media and files to the media manager service. What should you do? (Each correct answer presents part of the solution. Choose all that apply.)

- A. Insert the following line of code at line AP28.

```
IReadOnlyList<UploadOperation> upload =  
await BackgroundUploader.GetCurrentUploadsAsync();
```

- B. Insert the following line of code at line AP28.

```
UploadOperation upload = await uploader.CreateUpload(uri, uploadGrp);
```

- C. Insert the following line of code at line AP28.

```
UploadOperation upload = await uploader.CreateUploadAsync(uri, uploadGrp);
```

- D. Insert the following line of code at line AP19.

```
IReadOnlyList<StorageFile> files = await picker.PickMultipleFilesAsync  
();
```

- E. Insert the following line of code at line AP19.

```
IReadOnlyList<StorageFile> files = await picker.PickSingleFilesAsync  
();
```

- A. Option A
- B. Option B
- C. Option C
- D. Option D
- E. Option E

**Correct Answer:** BD

### QUESTION 5

#### DRAG DROP

You need to implement the photo viewer page to meet the business requirements.

How should you complete the code segment?

To answer, drag the appropriate [source or sources] to the correct location or locations in the answer area.

<div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">&lt;RepositionThemeTransition/&gt;</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">&lt;AddDeleteThemeTransition/&gt;</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">&lt;ReorderThemeTransition/&gt;</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">&lt;EntranceThemeTransition/&gt;</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">&lt;ViewBox/&gt;</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">&lt;GridView/&gt;</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">&lt;FlipView/&gt;</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">&lt;WrapGrid/&gt;</div>	<p>Answer Area</p> <pre> &lt;Button Content="Add New Photo" Click="btnAdd_Click"/&gt; &lt;Button Content="Remove Selected Photo" Click="btnDelete_Click &lt;ItemsControl Grid.Row="1" x:Name="rectangleItems"&gt;   &lt;ItemsControl.ItemContainerTransitions&gt;     &lt;TransitionCollection&gt;       [ ]     &lt;/TransitionCollection&gt;   &lt;/ItemsControl.ItemContainerTransitions&gt;   &lt;ItemsControl.ItemsPanel&gt;     &lt;ItemsPanelTemplate&gt;       [ ]     &lt;/ItemsPanelTemplate&gt;   &lt;/ItemsControl.ItemsPanel&gt; &lt;/ItemsControl&gt;                 </pre>
--	--

#### Correct Answer:

<div style="border: 1px solid green; padding: 2px; margin-bottom: 2px;">&lt;RepositionThemeTransition/&gt;</div> <div style="border: 1px solid green; padding: 2px; margin-bottom: 2px;">&lt;AddDeleteThemeTransition/&gt;</div> <div style="border: 1px solid green; padding: 2px; margin-bottom: 2px;">&lt;ReorderThemeTransition/&gt;</div> <div style="border: 1px solid green; padding: 2px; margin-bottom: 2px;">&lt;EntranceThemeTransition/&gt;</div> <div style="border: 1px solid green; padding: 2px; margin-bottom: 2px;">&lt;ViewBox/&gt;</div> <div style="border: 1px solid green; padding: 2px; margin-bottom: 2px;">&lt;GridView/&gt;</div> <div style="border: 1px solid green; padding: 2px; margin-bottom: 2px;">&lt;FlipView/&gt;</div> <div style="border: 1px solid green; padding: 2px; margin-bottom: 2px;">&lt;WrapGrid/&gt;</div>	<p>Answer Area</p> <pre> &lt;Button Content="Add New Photo" Click="btnAdd_Click"/&gt; &lt;Button Content="Remove Selected Photo" Click="btnDelete_Click &lt;ItemsControl Grid.Row="1" x:Name="rectangleItems"&gt;   &lt;ItemsControl.ItemContainerTransitions&gt;     &lt;TransitionCollection&gt;       &lt;GridView/&gt;     &lt;/TransitionCollection&gt;   &lt;/ItemsControl.ItemContainerTransitions&gt;   &lt;ItemsControl.ItemsPanel&gt;     &lt;ItemsPanelTemplate&gt;       &lt;WrapGrid/&gt;     &lt;/ItemsPanelTemplate&gt;   &lt;/ItemsControl.ItemsPanel&gt; &lt;/ItemsControl&gt;                 </pre>
--	--

### QUESTION 6

You need to implement a custom control to display thumbnail images of video clips. Which code segment should you use?

- A. 

```
public sealed class DownloadedVideoList: FlipView
{
    public DownloadedVideoList()
    {
        this.DefaultStyleKey = typeof(ListView);
    }
}
```
- B. 

```
public sealed class DownloadedVideoList: FlipView
{
    public DownloadedVideoList()
    {
        this.DefaultStyleKey = typeof(DownloadedVideoList);
    }
}
```
- C. 

```
public sealed class DownloadedVideoList: ListView
{
    public DownloadedVideoList()
    {
        this.DefaultStyleKey = typeof(DownloadedVideoList);
    }
}
```
- D. 

```
public sealed class DownloadedVideoList: ListView
{
    public DownloadedVideoList()
    {
        this.DefaultStyleKey = typeof(ListView);
    }
}
```

- A. Option A  
B. Option B  
C. Option C  
D. Option D

**Correct Answer: C**

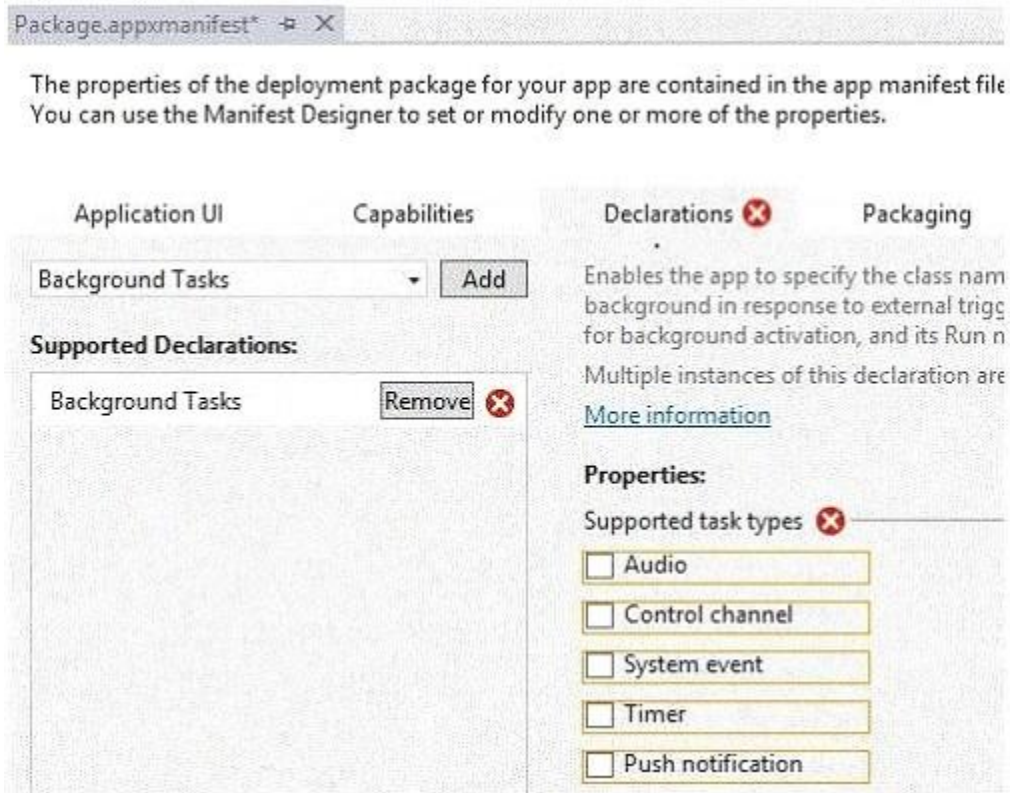
**QUESTION 7**

**HOTSPOT**

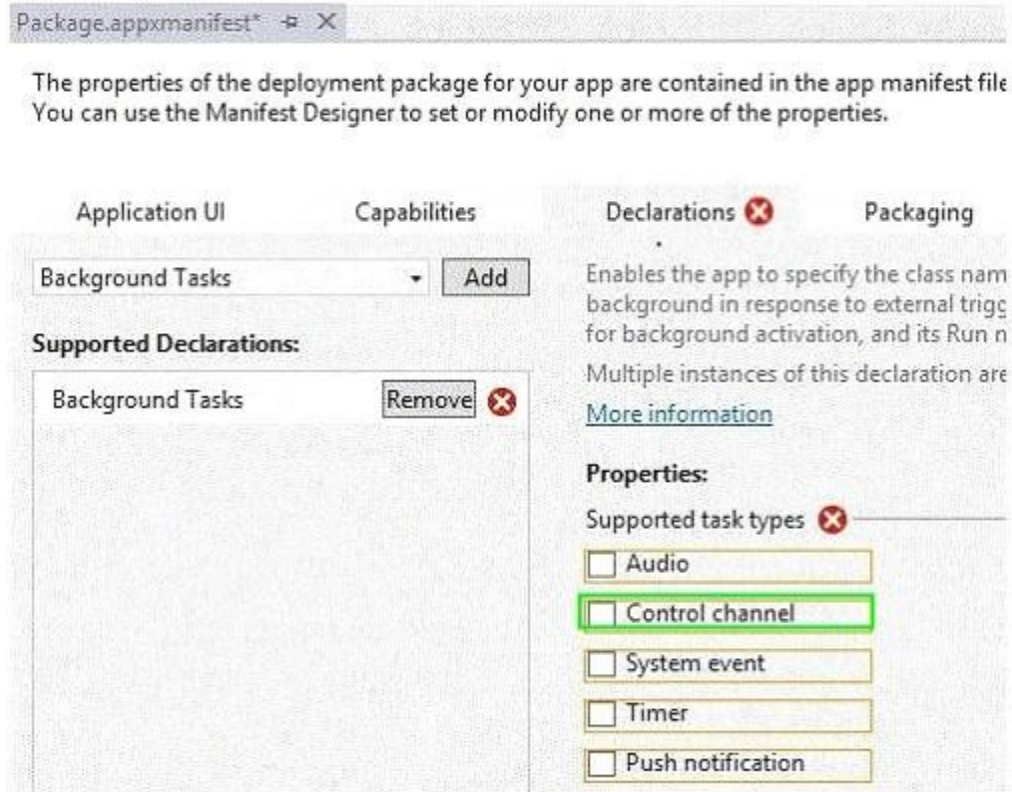
You need to configure the app manifest to support the file download requirements.

Which task type property should you specify?

To answer, select the appropriate property in the answer area.



**Correct Answer:**



**QUESTION 8**

You need to implement the requirements for the playback of media. What should you do? (Each correct answer presents part of the solution. Choose all that apply.)

- A. Add the following line of code at line MC02.  

```
private void ShowPlayTo()
{
    Windows.Media.PlayTo.PlayToManager.ShowPlayToUI();
}
```
- B. Add the following line of code at line MC06.  

```
ptMgr.DefaultSourceSelection = false;
```
- C. Add the following line of code at line MC10.  

```
ptMgr.PlayRequested += SourceRequestHandler;
```
- D. Add the following line of code at line MC05.  

```
ptMgr.SourceRequested += SourceRequestHandler;
```

**Correct Answer:** BD

### QUESTION 9

You need to implement the requirements for the behavior of the main page. Which code segment should you insert at line MP07?

- A. 

```
<VisualStateGroup x:Name="ApplicationViewStates">
  <VisualState x:Name="Snapped">
    <Storyboard>
      <ObjectAnimationUsingKeyFrames Storyboard.TargetName="LayoutRoot"
        Storyboard.TargetProperty="(Grid.RowDefinitions)[1].Height">
        <DiscreteObjectKeyFrame KeyTime="0" Value="Auto"/>
      </ObjectAnimationUsingKeyFrames>
    </Storyboard>
  </VisualState>
</VisualStateGroup>
```
- B. 

```
<VisualStateGroup x:Name="ApplicationViewStates">
  <VisualState x:Name="Filled">
    <Storyboard>
      <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="LayoutRoot">
        <DiscreteObjectKeyFrame KeyTime="0" Value="*" />
      </ObjectAnimationUsingKeyFrames>
    </Storyboard>
  </VisualState>
</VisualStateGroup>
```
- C. 

```
<VisualStateGroup x:Name="ApplicationViewStates">
  <VisualState x:Name="FullScreenLandscape">
    <Storyboard>
      <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="LayoutRoot">
        <DiscreteObjectKeyFrame KeyTime="0" Value="Auto" />
      </ObjectAnimationUsingKeyFrames>
    </Storyboard>
  </VisualState>
</VisualStateGroup>
```
- D. 

```
<VisualStateGroup x:Name="ApplicationViewStates">
  <VisualState x:Name="FullScreenPortrait">
    <Storyboard>
      <ObjectAnimationUsingKeyFrames Storyboard.TargetName="LayoutRoot"
        Storyboard.TargetProperty="(Grid.RowDefinitions).Height">
        <DiscreteObjectKeyFrame KeyTime="0" Value="*" />
      </ObjectAnimationUsingKeyFrames>
    </Storyboard>
  </VisualState>
</VisualStateGroup>
```

- A. Option A
- B. Option B
- C. Option C
- D. Option D

**Correct Answer: A**

**QUESTION 10**

**HOTSPOT**

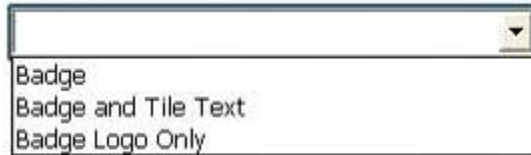
You need to meet the business requirements about downloading and uploading.

How should you configure the app?

To answer, select the appropriate options from each drop-down list in the answer area.

Configure the Application UI settings in Package.appxmanifest

Lock screen notifications:



A drop-down menu with a downward arrow on the right. The menu is open, showing three options: "Badge", "Badge and Tile Text", and "Badge Logo Only".

Logo files:



A drop-down menu with a downward arrow on the right. The menu is open, showing three options: "Badge Logo Only", "Wide Logo Only", and "Badge Logo and Wide Logo".

Configure the Declarations settings in Package.appxmanifest

Add a Background Task declaration and configure support for the

following task types:



A drop-down menu with a downward arrow on the right. The menu is open, showing five options: "Audio", "Control channel", "System event", "Timer", and "Push Notification".

**Correct Answer:**



Configure the Application UI settings in Package.appxmanifest

Lock screen notifications:

- Badge
- Badge and Tile Text
- Badge Logo Only

Logo files:

- Badge Logo Only
- Wide Logo Only
- Badge Logo and Wide Logo

Configure the Declarations settings in Package.appxmanifest

Add a Background Task declaration and configure support for the following task types:

- Audio
- Control channel
- System event
- Timer
- Push Notification

#### QUESTION 11

You need to ensure that the VideoProcessor component can be used by the Windows Store app. What should you do? (Each correct answer presents part of the solution. Choose all that apply.)

- A. Add the following attribute to line IP19.  
[Windows.Foundation.Metadata.DefaultOverload()]
- B. Replace line IP01 with the following line of code.  
Static class VideoProcessor
- C. Replace line IP09 with the following line of code.  
PublicVideoProcessor(string videoName, int ID)
- D. Add the following attribute to line IP14.  
[Windows.Foundation.Metadata.DefaultOverload()]
- E. Replace line IP01 with the following line of code.  
Public sealed class VideoProcessor

**Correct Answer:** ACE

### QUESTION 12

You need to implement the business requirements for providing information about file uploads and downloads. Which code segment should you use in the VideoProcessor.es class?

- A. 

```
public static IAsyncOperationWithProgress<TResult, TProgress> Run<TResult, TProgress>(
    Func<CancellationToken, IProgress<TProgress>, Task<TResult>> taskProvider)
{
    ...
}
```
- B. 

```
public static IAsyncActionWithProgress<TProgress> Run<TProgress>(
    Func<CancellationToken, IProgress<TProgress>, Task> taskProvider)
{
    ...
}
```
- C. 

```
public interface IAsyncOperation<TResult> : IAsyncInfo
{
    AsyncOperationCompletedHandler<TResult> Completed { get; set; }
    TResult GetResults();
}
```
- D. 

```
public interface IAsyncActionWithProgress<TProgress> : IAsyncInfo
{
    AsyncActionWithProgressCompletedHandler<TProgress> Completed { get; set; }
    AsyncActionProgressHandler<TProgress> Progress { get; set; }
    void GetResults();
}
```
- E. 

```
public static IAsyncOperation<TResult> Run<TResult>(
    Func<CancellationToken, Task<TResult>> taskProvider)
{
    ...
}
```

- A. Option A  
 B. Option B  
 C. Option C  
 D. Option D

**Correct Answer:** A

### QUESTION 13

You need to implement the requirements for streaming media. What should you do? (Each correct answer presents part of the solution. Choose all that apply.)

- A. Enable access to the Videos Library.  
 B. Ensure that the app stays in the foreground while media is being streamed.  
 C. Enable access to the Pictures Library.  
 D. Register for the SourceRequested event.  
 E. Enable access to the Music Library.  
 F. Register for the PlayRequested event.

**Correct Answer:** AD

**Explanation:**

From scenario:

Team members must be able to stream video clips to other devices in the vicinity of the team member's device. The app will not support the streaming of photographs.

D: You can use Play To to stream the audio or video in your application, as well as images, by implementing the Play To contract. To implement the Play To contract in your application, register for the sourceRequested event.

Note:

To register for the sourceRequested event, get a reference to the current PlayToManager by calling the getForCurrentView method. You can then call addEventHandler on the PlayToManager to associate your event handler with the sourceRequested event. In your event handler, pass the media element from your application to the setSource method of the PlayToSourceRequestedEventArgs object passed to the event handler as shown in the following example.

```
// Play To Contract
```

```
private Windows.Media.PlayTo.PlayToManager ptm =  
Windows.Media.PlayTo.PlayToManager.GetForCurrentView();  
  
protected override void OnNavigatedTo(NavigationEventArgs e) {  
ptm.SourceRequested += sourceRequestHandler;  
}  
  
private void sourceRequestHandler(  
  
Etc.
```

**QUESTION 14**

You need to implement the behavior requirements for the photo viewer. Which controls should you create?

- A. Create two SemanticZoom controls and one ListView control.
- B. Create one SemanticZoom control and one ListView control.
- C. Create one ScrollViewer control, one SemanticZoom control, and one GridView control.
- D. Create two GridView controls and one SemanticZoom control.

**Correct Answer: D**

## Topic 2, Scenario Geese

### Background

You are developing a Windows Store app. The app will allow ornithologists to photograph migrating geese, taking note of the location, heading, and weather conditions at the time each photo is taken.

### BusinessRequirements

The app must adhere to the following requirements:

- Create and store photographs of migrating geese.
- Record the location and weather conditions where the photograph was taken.
- Record the heading and time that the photograph was taken.
- Allow the user to display the information on any device that supports the PlayTo feature.

### TechnicalRequirements

#### General

The app must meet the following technical requirements:

- The app must store images and image metadata in the Pictures Library.
- The metadata logic must be encapsulated within a reusable component named LogicComponent1.
- The metadata logic must be available to Windows Store apps written in Visual Basic, C#, JavaScript, and C++.

#### Hardware

The app requires a device with camera, compass, and GPS features.

The app requires a device with Internet capabilities.

**CurrentEnvironment.cs**

```

CE01 namespace CurrentEnvironment
CE02 {
CE03     public sealed class Environment
CE04     {
CE05         private Compass _compass = null;
CE06         private LightSensor _light = null;
CE07         public IAsyncOperation<EnvironmentalStatus> GetCurrentEnvironmentAsync()
CE08         {
CE09             LoadSensors();
CE10             return (IAsyncOperation<EnvironmentalStatus>) AsyncInfo.Run(
CE11                 (System.Threading.CancellationToken) =>
InternalGetCurrentEnvironmentAsync());
CE12         }
CE13
CE14         private async Task<EnvironmentalStatus>
InternalGetCurrentEnvironmentAsync()
CE15         {
CE16             EnvironmentalStatuses = new EnvironmentalStatus();
CE17             es.Location = await GetLocationAsync();
CE18             ...
CE19             es.Temperature = await GetWeatherAsync();
CE20             es.Time = DateTime.UtcNow.ToString();
CE21
CE22             return es;
CE23         }
CE24
CE25         private async Task<string> GetLocationAsync()
CE26         {
CE27             var locator = new Geolocator();
CE28             Geoposition location = await locator.GetGeopositionAsync();
CE29             string curPosition = location.Coordinate.Latitude.ToString() + ", "
CE30                 + location.Coordinate.Longitude.ToString();
CE31             if (_compass != null)
CE32                 curPosition += ", " + _compass.GetCurrentReading
() .HeadingTrueNorth.Value;
CE33             return curPosition;
CE34         }
CE35
CE36         private async Task<string> GetWeatherAsync()
CE37         {
CE38             IList<WeatherData> weatherData = GooseLogic.GetWeatherData();
CE39         }
CE40
CE41         private void LoadSensors()
CE42         {
CE43
CE44             {
CE45                 _compass = Compass.GetDefault();
CE46             }
CE47         }
CE48     }
CE49
CE50     public struct EnvironmentalStatus
CE51     {
CE52         public string Location;
CE53         public string Time;
CE54         public string Temperature;
CE55     }
CE56 }

```

**MainPage.xaml.cs**

```
MP01 privateasyncvoidCapturePhoto_Click(objectsender, RoutedEventArgse)
MP02 {
MP03     try
MP04     {
MP05         CameraCaptureUIcameraUI = newCameraCaptureUI();
MP06         SizeaspectRatio = newSize(16, 9);
MP07         cameraUI.PhotoSettings.CroppedAspectRatio = aspectRatio;
MP08
MP09         StorageFilefile = awaitcameraUI.CaptureFileAsync
MP10         (CameraCaptureUIMode.Photo);
MP11         if(file != null)
MP12         {
MP13             varnewFile =
MP14             awaitWindows.Storage.KnownFolders.PicturesLibrary.CreateFileAsync(file.Name);
MP15             awaitfile.CopyAndReplaceAsync(newFile);
MP16             BitmapImagebitmapImage = newBitmapImage();
MP17             using(IRandomAccessStreamfileStream = awaitnewFile.OpenAsync
MP18             (FileAccessMode.Read))
MP19             {
MP20                 bitmapImage.SetSource(fileStream);
MP21             }
MP22             capturedPhoto.Source = bitmapImage;
MP23
MP24             varenv = newCurrentEnvironment.Environment();
MP25             varenvData = awaitenv.GetCurrentEnvironmentAsync();
MP26
MP27             Info.Text = envData.Location;
MP28         }
MP29     else
MP30     {
MP31         Info.Text = "An error has occurred";
MP32     }
MP33 }
MP34 catch(Exceptionex)
MP35 {
MP36     ...
MP37 }
```

**Package.appxmanifest**

```

PA01 <?xmlversion="1.0"encoding="utf-8"?>
PA02 <Packagexmlns="http://schemas.microsoft.com/appx/2010/manifest">
PA03 <IdentityName="7d32c109-5e1d-432a-a53f-df00440658f0"Publisher="CN=Admin"
Version="1.0.0.0"/>
PA04 <Properties>
PA05 <DisplayName>GooseTracker</DisplayName>
PA06 <PublisherDisplayName>Admin</PublisherDisplayName>
PA07 <Logo>Assets\StoreLogo.png</Logo>
PA08 </Properties>
PA09 <Prerequisites>
PA10 <OSMinVersion>6.2.1</OSMinVersion>
PA11 <OSMaxVersionTested>6.2.1</OSMaxVersionTested>
PA12 </Prerequisites>
PA13 <Resources>
PA14 <ResourceLanguage="x-generate"/>
PA15 </Resources>
PA16 <Applications>
PA17 <ApplicationId="App"Executable="$targetnametoken$.exe"
EntryPoint="GooseTracker.App">
PA18 <VisualElementsDisplayName="GooseTracker"Logo="Assets\Logo.png"
SmallLogo="Assets\SmallLogo.png"
Description="GooseTracker"ForegroundText="light"
BackgroundColor="#464646">
PA19 <DefaultTileShowName="allLogos"/>
PA20 <SplashScreenImage="Assets\SplashScreen.png"/>
PA21 </VisualElements>
PA22 </Application>
PA23 </Applications>
PA24 <Capabilities>
PA25
PA26 <CapabilityName="internetClient"/>
PA27 <DeviceCapabilityName="webcam"/>
PA28 <DeviceCapabilityName="location"/>
PA29 </Capabilities>
PA30 </Package>

```

**GooseTracker.csproj**

```

GO01 <ProjectToolsVersion="4.0"DefaultTargets="Build"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
GO02
GO03 <ItemGroup>
GO04
GO05 </ItemGroup>
GO06 </Project>

```

**GoosePlayTo.cs**

```
PT00 publicclassGoosePlayTo
PT01 {
PT02     privateWindows.Media.PlayTo.PlayToManagerplayToManager;
PT03     privateWindows.UI.Core.CoreDispatcherdispatcher;
PT04     privateMediaElementelement;
PT05
PT06     publicGoosePlayTo(MediaElementelement)
PT07     {
PT08         dispatcher = Window.Current.CoreWindow.Dispatcher;
PT09         playToManager = Windows.Media.PlayTo.PlayToManager.GetForCurrentView
PT10         ();
PT11         playToManager.SourceRequested += SourceRequested;
PT12         this.element = element;
PT13     }
PT14     privatevoidSourceRequested(Windows.Media.PlayTo.PlayToManagersender,
PT15     Windows.Media.PlayTo.PlayToSourceRequestedEventArgsargs)
PT16     {
PT17         vardef = args.SourceRequest.GetDeferral();
PT18         varevthander = dispatcher.RunAsync
PT19         (Windows.UI.Core.CoreDispatcherPriority.Normal,
PT20         () =>
PT21         {
PT22             args.SourceRequest.SetSource(element.PlayToSource);
PT23             def.Complete();
PT24         });
PT25     }
PT26
PT27     private asyncvoidLoadFile(Windows.Storage.StorageFilevideoFile,
stringcontentType)
PT28     {
PT29         varstream = awaitvideoFile.OpenAsync
PT30         (Windows.Storage.FileAccessMode.Read);
PT31     }
PT32
PT33     privatevoidPlay()
PT34     {
PT35         element.Play();
PT36     }
PT37
PT38     privatevoidPause()
PT39     {
PT40         element.Pause();
PT41     }
PT42 }
```



## Camera.cs

```
CA01 publicclassCamera: Windows.Media.Devices.IMediaDeviceController
CA02 {
CA03     privateWindows.Media.Capture.MediaCapturemedia;
CA04     privateWindows.Media.Devices.VideoDeviceControllervideo;
CA05     publicdoubleWhiteBalance
CA06     {
CA07         get
CA08         {
CA09             doublewbValue = -1.0;
CA10
CA11             returnwbValue;
CA12         }
CA13     }
CA14
CA15     publicboolSupportsBacklightCompensation
CA16     {
CA17         get
CA18         {
CA19
CA20         }
CA21     }
CA22
CA23     publicCamera()
CA24     {
CA25         media = newWindows.Media.Capture.MediaCapture();
CA26         ...
CA27         video = media.VideoDeviceController;
CA28
CA29
CA30     }
CA31 }
```

**QUESTION 15**

You need to register the reusable WinMD component. What should you do?

- A. In GooseTracker.csproj, add the following code at line G004.

```
<ProjectReference Include="..\LogicComponent1\LogicComponent1.csproj">
  <Project>{b64bd7c9-fbdc-4b80-8350-8fead0878721}</Project>
  <Name>GooseLogic</Name>
</ProjectReference>
```

- B. In the MainPage.xaml.cs file, register the handler for the extension/mime-type.

- C. Run the **Gacutil.exe /I shared.dll** command.

- D. In Package.appxmanifest, add the following code immediately after line G002.

```
<Extension Include="..\LogicComponent1\LogicComponent1.csproj">
  <Project>{b64bd7c9-fbdc-4b80-8350-8fead0878721}</Project>
  <Name> GooseLogic</Name>
</Extension>
```

- A. Option A  
B. Option B  
C. Option C  
D. Option D

**Correct Answer: A**

## EnsurePass.com Members Features:

1. Verified Answers researched by industry experts.
2. Q&As are downloadable in PDF and VCE format.
3. 98% success Guarantee and **Money Back** Guarantee.
4. Free updates for **180** Days.
5. **Instant Access to download the Items**

View list of All Exam provided:

<http://www.ensurepass.com/certifications?index=A>

To purchase Lifetime Full Access Membership click here:

<http://www.ensurepass.com/user/register>

**Valid Discount Code for 2015: JREH-G1A8-XHC6**

To purchase the HOT Microsoft Exams:

<u>Microsoft</u>			
<a href="#">70-243</a>	<a href="#">70-347</a>	<a href="#">70-466</a>	<a href="#">70-515</a>
<a href="#">70-246</a>	<a href="#">70-410</a>	<a href="#">70-467</a>	<a href="#">70-516</a>
<a href="#">70-247</a>	<a href="#">70-411</a>	<a href="#">70-480</a>	<a href="#">70-519</a>
<a href="#">70-321</a>	<a href="#">70-412</a>	<a href="#">70-483</a>	<a href="#">70-583</a>
<a href="#">70-331</a>	<a href="#">70-413</a>	<a href="#">70-484</a>	<a href="#">70-640</a>
<a href="#">70-332</a>	<a href="#">70-414</a>	<a href="#">70-485</a>	<a href="#">70-649</a>
<a href="#">70-336</a>	<a href="#">70-417</a>	<a href="#">70-486</a>	<a href="#">70-668</a>
<a href="#">70-337</a>	<a href="#">70-461</a>	<a href="#">70-487</a>	<a href="#">70-680</a>
<a href="#">70-341</a>	<a href="#">70-462</a>	<a href="#">70-488</a>	<a href="#">70-687</a>
<a href="#">70-342</a>	<a href="#">70-463</a>	<a href="#">70-489</a>	<a href="#">70-688</a>
<a href="#">70-346</a>	<a href="#">70-464</a>	<a href="#">70-513</a>	<a href="#">70-689</a>

