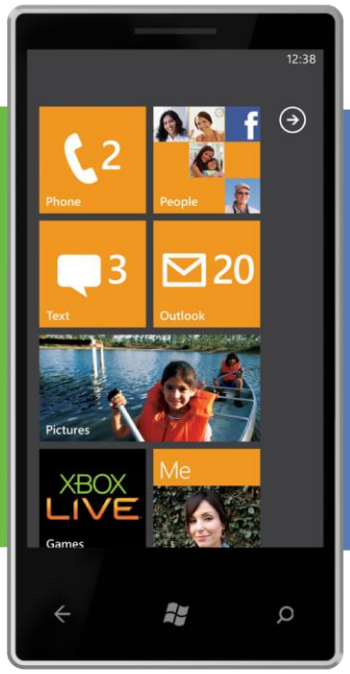
Windows Phone

Windows Phone 7 Application Certification Requirements



7

Microsoft Corporation

June 7, 2010

Version 1.0

Applies To: Windows® Phone 7

Contents

Windows Phone 7 Application Certification Requirements

The purpose of this document is to provide the policies and technical requirements that a Windows® Phone 7 application or game must meet to pass certification and to be eligible for listing in Windows Phone Marketplace.

1.0 Program Overview

A core principle that is applied in designing the certification process is that each individual policy or requirement is clear, objective, and testable. This transparency is designed to help developers easily design and test applications to meet these requirements.

The following list shows the pillars of the certification program:

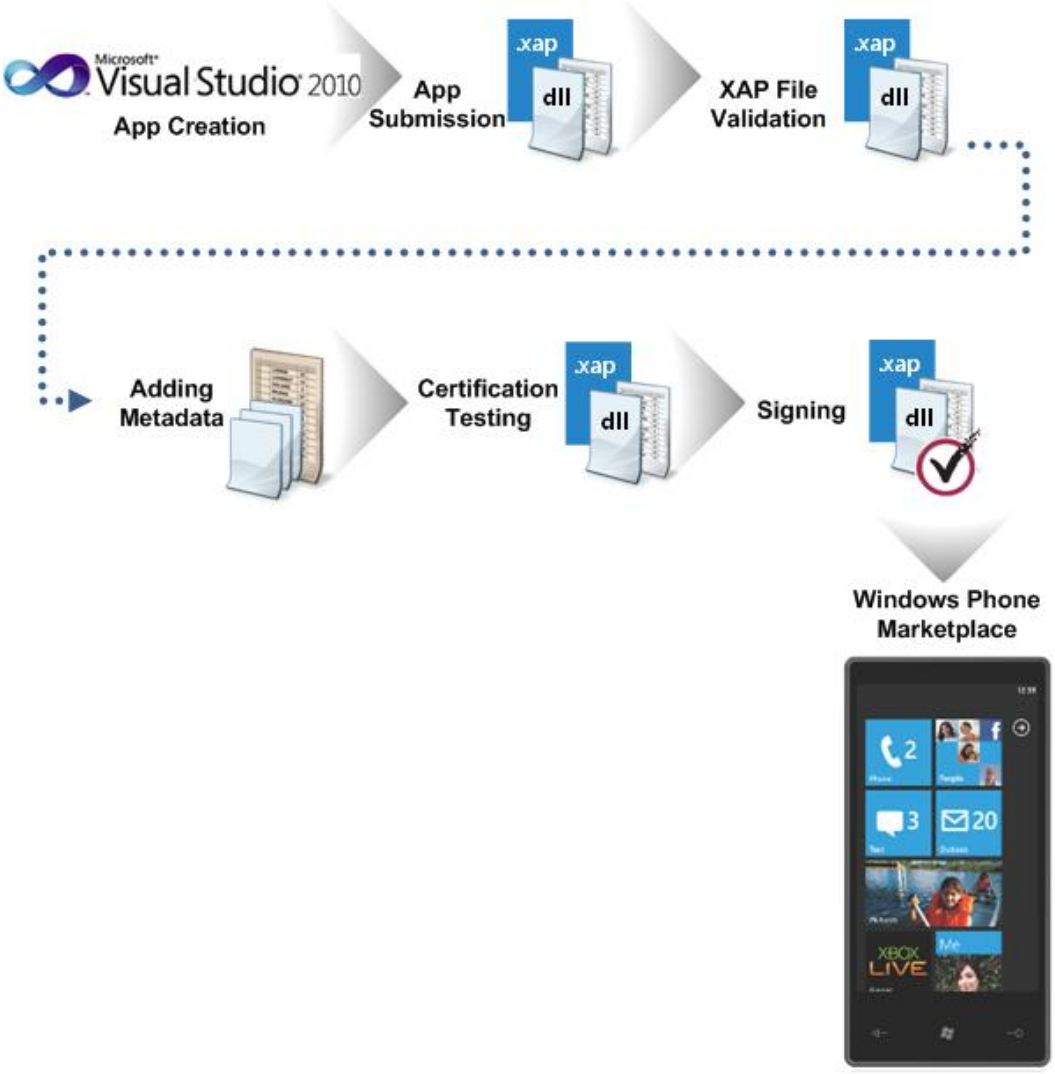
1. Applications are reliable.
2. Applications make efficient use of resources.
3. Applications do not interfere with the phone functionality.
4. Applications are free of malicious software.

1.1 What You Need to Know About the Submission and Certification Process

When your application is ready for publication, it must go through the certification process before it is eligible for listing in Windows Phone Marketplace. Your application does not have to be signed before submission.

The certification process involves static validation and automated testing of your application to verify that it meets all the policies and requirements. The following list shows the five major categories of policies and requirements:

The following is a simplified illustration of the submission and certification process.



1.1.1 Process Outline

The following is a brief outline of the submission and certification process:

1. Sign in to your account in the Windows Phone 7 developer portal.
2. Create a new application submission.
3. Upload the application XAP file.

4. Enter the metadata for the application, such as description, category, and iconography.
5. Select the distribution countries and pricing.
6. The XAP file is validated while you are entering metadata.
7. If the XAP file validation succeeds, the submission process continues to Step 8; otherwise, the process terminates and you get a notification.
8. Select the option to publish immediately after passing the certification process or to wait until you decide to publish.
9. The XAP file is repackaged as described in Section 4.1.2.
10. The repackaged XAP is file is deployed to a phone for the certification testing. Certification involves the automated and manual verification of the meeting of the requirements that are described in Sections 2, 3, 4, 5, and 6.
11. If the application meets all the requirements, the repackaged XAP and assembly files are signed, and the application is eligible for publication according to the option selected in Step 8.

    If the application fails one or more of the requirements, you get a failure report and the application is not published.

Note:

When you submit an application update for certification, it goes through the same process as the original application.

1.1.2 Code Signing

Code signing occurs automatically once the application has successfully passed the certification testing without any failure. The application and repackaged XAP files are signed with the Authenticode® certificate assigned to you when you registered on Windows Phone Marketplace. Any signatures in a submitted application or XAP files will be replaced and are not retained.

Note:

All applications must be signed with the Microsoft issued Authenticode certificate before they can be installed and run on commercially available Windows Phones.

2.0 Application Policies

To protect the Windows Phone Marketplace service and users of the service, and to address mobile operator requirements, Microsoft has established the following policies for applications offered for distribution in the Windows Phone Marketplace. Microsoft reserves the right to update these policies as needed.

2.1 Your application must be fully functional when acquired from the Windows Phone Marketplace (except for additional data as permitted below), and may not require the user to pay outside of Windows Phone Marketplace to activate, unlock, upgrade, or extend usage of the application.

2.2 Your application may not sell, link to, or otherwise promote mobile voice plans.

2.3 Your application may not consist of, distribute, link to, or incent users to download, or otherwise promote alternate marketplaces for applications and/or games.

2.4 Your application must not jeopardize the security or functionality of (a) Windows Phone 7 devices or (b) the Windows Phone Marketplace.

2.5 The OTA (over the air) installation file for the application may not exceed 20 MB. Applications in excess of that size will be downloaded via Wi-Fi or through a tethered connection to a PC running the appropriate Microsoft software.

2.6 If your application includes a trial version, the trial version must reasonably represent the functionality and quality of the full application.

2.7 If your application includes or displays advertising, the advertising must comply with the Microsoft Advertising Creative Acceptance Policy Guide (http://advertising.microsoft.com/creative-specs).

2.8 If your application requires the download of a large additional data package (e.g. >50 MB) to enable the application to run as described, the application description must disclose the approximate size of the data package and that additional charges may apply depending on connectivity used to acquire data.

2.9 If your application enables chat, instant messaging, or other person-to-person communication and allows the user to setup or create his or her account or ID from the mobile device, the application must include a mechanism to verify that the user creating the account or ID is at least 13 years old.

2.10 If your application publishes a user's personal information from the mobile device to any service or other person, the application must implement "opt-in" consent. "Personal information" means all information or data associated with an identifiable user, including but not limited to the following, whether stored on the mobile device or on a web-based server that is accessible from the mobile device:

- Location information
- Contacts
- Photos
- Phone number
- SMS or other text communication
- Browsing history

To “implement ‘opt-in’ consent,” the application must (1) first describe how the personal information will be used or shared; (2) obtain the user’s express permission before publishing the information as described; and (3) provide a mechanism through which the user can later opt out of having the information published.

2.11 If your application allows users to purchase music content, it must include the Windows Phone music Marketplace (if available) as a purchase option. If the application also allows music content to be purchased from any source other than the Windows Phone music Marketplace, the application must include its own playback functionality for that music content.

2.12 If your application uses the Microsoft Push Notification Service (PNS), the application and the use of the PNS must comply with the following requirements:

- 2.12.1 The application must first describe the notifications to be provided and obtain the user’s express permission (opt-in), and must provide a mechanism through which the user can opt out of receiving push notifications. All notifications provided using PNS must be consistent with the description provided to the user and must comply with all applicable 2.0 Application Policies and 3.0 Content Policies.
- 2.12.2 The application and its use of the PNS must not excessively use network capacity or bandwidth of the PNS or otherwise unduly burden a Windows Phone or other Microsoft device or service with excessive push notifications, as determined by Microsoft in its reasonable discretion, and must not harm or interfere with any Microsoft networks or servers or any third party servers or networks connected to the PNS.
- 2.12.3 The PNS may not be used to send notifications that are mission critical or otherwise could affect matters of life or death, including without limitation critical notifications related to a medical device or condition. MICROSOFT EXPRESSLY DISCLAIMS ANY WARRANTIES THAT THE USE OF PNS OR DELIVERY OF PNS NOTIFICATION WILL BE UNINTERRUPTED, ERROR FREE, OR OTHERWISE GUARANTEED TO OCCUR ON A REAL-TIME BASIS.

3.0 Content Policies

To protect the Windows Phone Marketplace service and users of the service, and to address mobile operator requirements, Microsoft has established the following policies for content offered for distribution in the Windows Phone Marketplace. Application content that is included in or accessed from your application must comply with the following policies. Microsoft reserves the right to update these policies as needed.

Game applications that have been rated by an approved ratings board (currently ESRB, PEGI, and USK) may submit the ratings certificate for the application and provide the appropriate content descriptors. If a game application is submitted with a rating that is equal to or less restrictive than ESRB “T”, PEGI “12”, or USK “12”, it will be presumed to comply with these content policies.

3.1 Licensed Content, Name, Logo & Trademarks

Content allowed where:

- Content and application name are original or licensed.
- Copyrighted content that is used with permission. Use of branded items (logos/trademarks) has been approved by the brand owners.
- If an application depicts any mobile or wired telephone, handheld PDA, or any other data and voice communicator, it must be either generic or a Windows Phone device.
- It is the application provider’s responsibility to determine if the application provider has the right to use the chosen name, content, logos, copyright, trademarks, online services & API’s.

3.2 Illegal or Contemplates Harm

Content not allowed:

- Any content that is illegal under applicable local law, obscene, or indecent.
- Any content that depicts or encourages harm or violence against a person or animal in the real world.

3.3 Defamatory, Libelous, Slanderous, and Threatening

Content not allowed:

- Any content that is defamatory, libelous, slanderous, or threatening.
- Any content that facilitates or promotes content prohibited by these guidelines.

3.4 Hate Speech or Discriminatory

Content not allowed:

- Any content that advocates discrimination, hatred, or violence based on considerations of race, ethnicity, national origin, language, gender, age, disability, status as a veteran, religion, sexual orientation or expression, or that promotes organizations devoted to that purpose. Such content can include images or text that perpetuates a negative stereotype of a race, gender, sexual preference or religion. We are particularly inclined to act against content where there is evidence that the intent of posting was to harass, threaten, or insult an individual or group on one of these bases.

3.5 Alcohol, Tobacco, Weapons, and Drugs

Content not allowed:

- Any content that facilitates or promotes, whether directly or indirectly, the illegal (under applicable local law) or excessive sale or use of alcohol or tobacco products, drugs, or weapons is not allowed on any section/site, regardless of targeting.
- Any content that facilitates the use of weapons in the real world.

3.6 Adult Related Content

Content not allowed:

- Sex / Nudity – Images that are sexually suggestive or provocative (e.g. sexually provocative touching, bondage, masturbation); provocative images that reveal nipples, genitals, buttocks, or pubic hair.
- Content that a reasonable person would consider to be adult or borderline adult content (images, text, or audio).
- Content that generally falls under the category of pornography.
- Content that depicts or suggests prostitution.
- Content depicting sexual fetishes.
- Content of a sexual nature depicting children or animals.

3.7 Certain Types of Illegal Activity

Content not allowed:

- Any content that facilitates or promotes illegal gambling, illegal adult content and/or pornography, child pornography, bestiality, piracy, illegal online pharmacies, illegal drugs, or criminal or terrorist activities.
- Any content that instructs users how to make bombs or weapons, drugs, or solicits involvement in behavior that is violent or illegal under applicable local law.
- Unauthorized use of another entity’s intellectual property, including but not limited to: software, music, art, and other copyrighted, trademarked or patented materials or trade secrets.
- Any content that facilitates or promotes underage drinking, consumption of illegal drugs, or socially irresponsible behavior due to alcohol or drug consumption (e.g., drinking and driving).

3.8 Violence

Content not allowed:

- Realistic or gratuitous violence, including depictions of the following:
  - Decapitation, impaling, blood splatter/blood spurting/blood pooling, or gore
  - Exploding body parts
  - Guns/weapons pointed toward user/audience
  - Strangulation/choking
  - People or creatures on fire
  - Cruelty to animals
  - Audio of humans or animals suffering
- Involuntary or physically-resisted sexual interactions with violent or illicit overtones
- Rape, sexual assault
- Molestation, physical child abuse
- Requests or instructions to injure or otherwise harm a real-world person or group of people
- Glorification of crimes against humanity such as genocide and torture

3.9 Excessive Profanity

Content not allowed:

- Any content with the excessive use of profanity or adult language.

Note:

Microsoft reserves the right to update these content policies as needed to protect the Windows Phone Marketplace service or the users of the service.

4.0 Application Submission Requirements

The following requirements are validated during the submission process. The process involves the checking of the metadata and the validating of the XAP file that you upload. An application that does not meet one or more requirements fails the submission process.

4.1 Installation Package Validation

The assembly and data files must be packaged as an XAP file package. Visual Studio® 2010 Express for Windows Phone generates the necessary XAP package and manifest files.

- **4.1.1 List of Package Requirements**

  The maximum size of the XAP package file is 500 MB.

  The XAP package must contain the following:

  a. A valid Windows Phone application manifest file, named WMAppManifest.xml. For more information, see the Windows Phone Application Manifest File topic.

  b. A valid .NET application manifest file, named AppManifest.xml.

  c. The assembly files as specified in the AppManifest.xml file.

  d. The application icon that you want displayed on the phone **start experience**.

  e. The application tile image that you want displayed when the user pins the application to the quick launch area on the phone **start experience**.

  The following table shows the size and file type requirements for icon and tile images.

Required Icons in the XAP Package	Pixels	File Type
Application Icon	62 x 62	PNG
Application Tile Image	173 x 173	PNG

  Microsoft recommends that images support 262 DPI.

  Microsoft recommends that you limit the size of the XAP package to 20 MB to allow the application to be distributed over the Mobile Operator network. Additionally, Microsoft recommends that the application executable files be no larger than 1 MB.

- **4.1.2 XAP File Repacking Process**

  When you submit the XAP file to Windows Phone Marketplace, the file is decompressed, validated, and repackaged.

  Repackaging involves the following steps:

  a. The Windows Phone application manifest is provisioned with a product identifier for each application.

  b. The security capabilities are rediscovered and listed in the manifest.

  c. The phone experience hub is set in the Windows Phone manifest (e.g. Media + Video hub).

  d. The Digital Right Management header file is created and named WMAppPRHeader.xml.

  e. The original contents of the package, the updated Windows Phone application manifest, and the Digital Rights Management header file are compressed into a new XAP package.

4.2 Application Code Validation

The following list shows the code requirements:

- **4.2.1** You must develop your application using the documented APIs that are supported on the Windows Phone Application Platform. For more information, see the Class Library Reference for Windows Phone topic.
- **4.2.2** The application must not invoke native code via PInvoke or COM interoperability. If it does, it will fail the certification process.
- **4.2.3** The application must be compiled using retail configuration instead of debug. The application must not contain debugging symbols or output.
- **4.2.4** The application must not redistribute the Windows Phone assemblies.

4.3 Phone Capabilities Detection

When you create a Windows Phone project using Visual Studio 2010 Express for Windows Phone, the Windows Phone application manifest file is auto-generated. The manifest file is populated with phone capabilities when you call APIs that require specific security permissions or user disclosure. The phone capabilities listed on the application manifest file are displayed to the user during application purchase. In addition, operating system grants the security permissions to the application according to the capabilities listed on the manifest file. For more information, see the Windows Phone Application Manifest File topic.

The application submission process uses Microsoft intermediate language (MSIL) code analysis to detect phone capabilities. The phone capabilities detected on your application are listed on the Windows Phone application manifest file replacing existing capabilities, as described in Step b, Section 4.1.2.

Note:

You can submit an application with obfuscated code and the detection process still applies.

Note:

The capability detection process does not discover Windows Phone APIs invoked via .NET reflection. As a result, the application will not have the security permissions required to run properly and will result in a failure during certification.

Note:

The capability list can change across application updates. When you submit an application update for certification, it goes through the same process as the original application.

4.4 Language Validation

Every application is targeted to publish to at least one specific geographic market and language. You can target multiple markets and submit your application in multiple languages. The language detection process includes the evaluation of the metadata that is used to describe the application and the UI text that is used within the application.

The following list shows the supported languages in Windows Phone 7:

- English
- French
- Italian
- German
- Spanish

An application must be localized in at least one of the supported application languages on the Windows Phone Marketplace. You will be prompted to submit a description only for the languages listed above. The users see this description when browsing the Windows Phone Marketplace catalog.

4.5 Windows Phone Marketplace Iconography

For each application, you must submit one icon to represent your application in the Windows Phone Marketplace catalog. This icon must match closely the icon provided in the XAP package. Users see this icon when browsing the application catalog on the phone before making a purchase.

The following table lists the phone application icon requirements.

Icon	Description	Size (Pixels)	File Type
Device application icon for Windows Phone Marketplace catalog	mandatory	99 x 99 (small) 173 x 173 (large)	PNG
Desktop application icon for Windows Phone Marketplace catalog.	mandatory	200 x 200	PNG
Windows Phone Marketplace panoramic background art	optional	1426 x 800	PNG

Microsoft recommends that images support 262 DPI.

4.6 Application Screenshot

For each application, you must provide at least one or up to a maximum of eight screenshots. The users see this screenshot in the details page of the catalog before they make a purchase.

The screenshot should be a direct capture of the phone screen or emulator. Graphically-enhanced screenshots are not allowed.

The following table lists the size and file type requirements for application screenshots.

Screenshot	Size (Pixels)	File Type
Details page screenshot	480 x 800	PNG

5.0 Application Certification Requirements

An application that is submitted for certification must meet all of the following requirements. An application that does not meet one or more of the requirements fails the certification process.

Note:

The requirements apply equally to an application that implements game functionality, commonly referred to as a game.

5.1 Application Reliability

- **5.1.1 Multiple Devices Support**

  The application must run on any Windows Phone 7 device, regardless of model, screen size, keyboard hardware, and manufacturer.

- **5.1.2 Application Termination**

  The application must handle exceptions raised by the .NET Framework and not terminate unexpectedly. During the certification process, the application is monitored for unexpected termination. An application that terminates unexpectedly fails certification.

  When handling exceptions, an application must provide a user-friendly error message. You may present a message that is relevant to the context of the application or use a Microsoft provided error message. The application must continue to run and remain responsive to user input after the exception is handled.

  An application that displays generic or unhelpful error messages will fail certification.

- **5.1.3 Application Does not Hang**

  The application must not become unresponsive to user input because of an operation within the application. For time consuming activities such as downloading data over network connections, the application must display a visual progress indicator. When there is a visual progress indicator, you must implement a UI element that provides the user with an option to cancel the time consuming activity.

5.2 Performance and Resource Management

- **5.2.1 Launch Time**

  a. The application must render the first screen within 5 seconds after launch.

     To meet this requirement, Microsoft recommends that the application provides a splash screen image in a file called SplashScreenImage.jpg in the root of the XAP package.

  b. Within 20 seconds after launch, the application must be responsive to user input.

- **5.2.2 Application Responsiveness After Being Paused**

  The application must resume and be functional after being paused. The launch time must meet the requirements in Section 5.2.1.

- **5.2.3 Use of Back Button**

  a. Pressing the Back button from the first screen of an application must exit the application.

  b. Pressing the Back button must return the application to the previous page. If the current page displays a context menu or a dialog, the pressing of the Back button must close the menu or dialog and cancel the backward navigation to the previous page.

- **5.2.4 Use of Back Button in Games**

  a. Pressing the Back button from the first screen of a game must exit the application.

  b. During gameplay, pressing the Back button in games must present an in-game menu. This menu must offer the option to resume the game. Pressing the Back button while this menu is up must exit the game. Microsoft recommends that you save the user game state or warn them of possible progress loss before exiting the game).

  c. Outside gameplay (for example, when the user is viewing the options or help menu), pressing the Back button must return to the previous menu or page.

5.3 Phone Functionality

The application must not delay or prevent the ability of the user to initiate a call, answer an incoming call, or end a call. The application must not delay or prevent the ability of the user to send or receive SMS or MMS messages.

The application must not hang or terminate unexpectedly when there is an incoming phone call, SMS, or MMS message.

5.4 Security

- **5.4.1 Malicious Software Screening**

  The application must be free of viruses, malware, and any malicious software.

- **5.4.2 MSIL Type Safety Verification**

  Windows Phone 7 operating system implements multiple sandbox mechanisms to help protect the integrity of the device and the applications running on the device. The Common Language Runtime (CLR) on Windows Phone 7 relies on the type-safe execution of application code to help enforce security and isolation mechanisms.

  An application must implement type-safe MSIL code to pass certification. For more information about C# unsafe code, see Unsafe Code and Pointers (C# Programming Guide.

- **5.4.3 Security Transparency Verification**

  The Windows Phone Application Platform does not allow an application to run security critical code. An application that invokes security critical code will fail certification.

  For more information about the .NET Security model, see Security Changes in the .NET Framework 4.

5.5 Content Validation

Content validation includes the evaluation of the metadata that describes the application as well as the UI text and media within the application. For example, an application that is submitted in French requires a product description in French. In addition to evaluating the description, the certification process reviews UI text, screenshots, and icons, in French.

Additional content validation criteria are described in Section 3.0. The application content (e.g. text, visual elements) must be visible and legible regardless of the phone theme. For example, if the phone theme changes from black background to white background, the text and visual elements of your application must be visible or legible.

When you submit separate XAP packages for each language, each XAP package is evaluated as individual submissions. By submitting a single XAP package for all supported languages you reduce the certification feedback time.

6.0 Additional Requirements for Specific Application Types

An application that uses specific phone capabilities must meet the following requirements in addition to all the requirements specified above.

6.1 Location Aware Application

A location aware application can access the phone location by using the classes in System.Devices.Location namespace. For more information, see the Location for Windows Phone topic.

Users have the ability to turn off the Location Service on the phone from the System Settings page. Location aware applications must not hang or terminate unexpectedly when the Location Service is turned off on the phone. One recommendation is to present a user friendly message to indicate that location data is not available. Another recommendation is to provide the user with the ability to view and disable the use of location data.

6.2 Push Notifications Application

The Microsoft Push Notification Service provides a dedicated, resilient, and persistent channel for pushing notifications from a web service to a mobile device. For more information, see the Push Notifications for Windows Phone topic in MSDN®.

An application that binds a push notification channel to a tile or toast must meet the following requirements:

- **6.2.1 Configurable Functionality**

  In the UI or settings menu, the application must provide the user with the ability to independently disable toast or tile notifications.

- **6.2.2 Initial Push Notifications Functionality**

  On first use of HttpNotificationChannel.BindtoShellToast or HttpNotificationChannel.BindToShellTile methods, the application must ask the user for explicit permission to receive Push Notification Service.

6.3 Applications Running under a Locked Screen

An application in the foreground can continue to run when the phone screen is locked by invoking the Microsoft.Phone.Shell.PhoneApplicationService.ApplicationIdleDetectionMode method.

An application that is running under a locked screen must meet the following requirements:

- **6.3.1 Configurable Functionality**

  In the UI or settings menu, an application must provide the user with the ability to prevent the application from running under a locked screen.

- **6.3.2 Launch Configuration of Running under a Locked Screen**

  On first use of ApplicationIdleDetectionMode, an application must ask the user for explicit permission to run under a locked screen.

Microsoft recommends that these applications minimize power usage as much as possible while running under a locked screen.

6.4 Music + Video Hub Application

An application in the Media + Video hub provides an integrated music and video experience on the phone as its primary function. An application that calls the Microsoft.Devices.MediaHistory or Microsoft.Devices.MediaHistoryItem classes is considered as a Music + Video hub application. The submission process detects that the application uses these classes and automatically updates the hub type to Music + Video in the Windows Phone application manifest.

An application in the Music + Video hub must meet the following requirements:

- **6.4.1** The application functionality must be related to audio and/or music media playback.
- **6.4.2** The application must launch the playback experience when the user taps on a hub tile in the “History” or “Now playing” area of the Music + Video hub.

  An application in the Music + Video hub may override music already playing in the background.

  Note:

  Requirements in Section 6.5 do not apply to Music + Video hub applications.

- **6.4.3** The application must update the “History” area of the Music + Video hub when the application plays media.
- **6.4.4** The application must update the “New” area of the Music + Video hub when media is added to the device or when the user creates an “object” in the application (for example, a radio station is created, a music tag is created.)
- **6.4.5** The application must update the “Now playing” area in the Music + Video hub when the application plays media. The “Now playing” hub tile must update for each song, video, or any other type of media that plays. “Now playing” must not represent a container object, such as a playlist, album, or TV series.

- **6.4.6** When the media is associated with a container, the hub tile in “New” and “History” in the Music + Video hub must represent a valid container, such as album, artist, playlist, radio station, rather than individual media items.
- **6.4.7** The hub tiles in the Music + Video hub must not contain advertisements, media feeds, or other unsolicited content.
- **6.4.8** Comply with the iconography rules for the Music + Video hub.

6.5 Applications that Play Music

An application can play music in the background, even when its primary function is not about music and video experience. An application that plays music must meet the following requirements:

- **6.5.1 Initial Launch Functionality**

  When the user is already playing music on the phone when the application is launched, the application must not pause, resume, or stop the active music in the phone MediaQueue by calling the Microsoft.Xna.Framework.Media.MediaPlayer class.

  If the application plays its own background music or adjusts background music volume, it must ask the user for consent to stop playing/adjust the background music (e.g. message dialog or settings menu).

  Note:

  This requirement does not apply to applications that play sound effects through the Microsoft.Xna.Framework.Audio.SoundEffect class, as sound effects will be mixed with the MediaPlayer. The SoundEffect class should not be used to play background music.

  Note:

  This requirement does not apply to Music + Video hub applications that are described in Section 6.4

- **6.5.2 Configurable Functionality**

  If the application needs to play its own background music or adjust background music volume, it must provide the user with the configurable settings for its music.

- **6.5.3 Applications that Play a Video or Audio Segment**

  If the application plays a non-interactive full motion video or audio segment (e.g. cut-scene or media clip), it should use the Microsoft.Phone.Tasks.MediaPlayerLauncher class to play the sequence and does not need to ask for user consent. When the media sequence is completed, it must resume the music by calling Microsoft.Xna.Framework.Media.MediaPlayer.Resume.

6.6 Photo Extras

Photo extras allow you to integrate your photo altering application with the Windows Phone built-in photo application. Through the extras menu in the single photo viewer, end users can access your application without leaving the primary Windows Phone photo application. You enable the “extras” menu in the single photo viewer of the phone by creating an Extras.xml file in the root of the XAP package. For more information, see the Photo Extras Application Extensibility for Windows Phone topic.

An application with Extras.xml in the root of the XAP package must meet the following requirements:

- **6.6.1** The application must implement the primary functionality associated with photo manipulation.
- **6.6.2** The Extras.xml file in the root XAP package must be valid according to the description in the Photo Extras Application Extensibility for Windows Phone topic.
- **6.6.3** The application must support two kinds of launching behavior.
    a. When an application is launched from the application list without an input photo, the application must invoke the PhotoChooser method to enable the user to pick up a photo or capture a new one.
    b. When an application is launched from the “extras” menu in the Single Photo Viewer with an input photo in the JPEG file format, the application must implement a functionality to allow the user to manipulate the input photo without any photo selection steps.

Microsoft recommends that the application notifies the user when it saves the edited photo.

7.0 Appendix: Windows Phone Application Manifest File

Overview

Windows Phone projects have an auto-generated XML manifest file that contains phone application metadata. The XML manifest file includes information such as product IDs, versioning details, runtime types, paths to resources, phone capabilities, and other application-specific information. The file is automatically updated every time you build or deploy your application. The primary purpose of this file is the following:

- The Windows Phone Marketplace application submission process utilizes information from the manifest file. The manifest file supports the submission of applications to the Windows Phone Marketplace (including certification), device marketplace filtering, marketplace-to-device deployment, and device execution.
- The information from the manifest file is used as the application metadata that will be stored in the application database.

In Windows Phone, this manifest file is called **WMAppManifest.xml**, and is visible in the **Properties** folder in **Solutions Explorer**. This section details the Windows Phone application manifest file schema for both Silverlight and XNA Framework application development. For the most part, both manifest files for Silverlight and XNA Framework applications are the same other than a few attribute value differences. These differences are detailed below as needed.

Important:

The application manifest file is automatically generated in Visual Studio and you should not edit the file manually. Your application may become unstable or unusable if you modify the contents of this file. Also, some of the values in the manifest file will be updated or changed after you submit your application to the Windows Phone Marketplace.

The following is an example of a Windows Phone application manifest file for a Silverlight project:


```
<?xml version="1.0" encoding="utf-8"?>

<Deployment xmlns="http://schemas.microsoft.com/windowsphone/2009/deployment"
AppPlatformVersion="7.0">
  <App xmlns="" ProductID=" {00000000-0000-0000-0000-000000000000}
Title="WindowsPhoneApplication" RuntimeType="Silverlight" Version="1.0.0.0"
Genre="NormalApp"  Author="" Description="" Publisher="">
    <IconPath IsRelative="false" IsResource="true">ApplicationIcon.png</IconPath>
    <Capabilities>
      <Capability Name="ID_CAP_NETWORKING" />
      <Capability Name="ID_CAP_LOCATION" />
      <Capability Name="ID_CAP_SENSORS" />
      <Capability Name="ID_CAP_MICROPHONE" />
      <Capability Name="ID_CAP_MEDIALIB" />
      <Capability Name="ID_CAP_GAMERSERVICES" />
      <Capability Name="ID_CAP_PHONEDIALER" />
      <Capability Name="ID_CAP_PUSH_NOTIFICATION" />
      <Capability Name="ID_CAP_WEBBROWSERCOMPONENT" />
    </Capabilities>
    <Tasks>
      <DefaultTask  Name ="_default" NavigationPage="Default task"/>
    </Tasks>
    <Tokens>
      <PrimaryToken TokenID="Token" TaskName="_default">
        <TemplateType5>
          <BackgroundImageURI IsRelative="true"
IsResource="false">Background.png</BackgroundImageURI>
          <Count>0</Count>
          <Title>WindowsPhoneApplication</Title>
        </TemplateType5>
      </PrimaryToken>
    </Tokens>
  </App>
</Deployment>
```


Note:

The example above contains a **Capabilities** element that provides the full list of the capabilities supported in Windows Phone. Visual Studio will populate the necessary capabilities when you build a project. For more information about the capability function, see the **Capabilities: <Capabilities></Capabilities>** section.

Windows Phone Manifest File Structure

This section defines and discusses the elements and attributes for the Windows Phone Silverlight and XNA Framework manifest files.

Deployment: < Deployment ></ Deployment >

This root element provides application part and localization information in the application manifest when deploying a Windows Phone application.

Attribute	Definition
xmlns	The XML namespace. The default value is "http://schemas.microsoft.com/windowsphone/2009/deployment"
AppPlatformVersion	Type: String. The version of the Windows Phone Developer Tools or the runtime binaries of the platform. The default value is 7.

Application: <App></App>

The **Application** element is a child element of the **Deployment** element and supplies information about items such as the product ID, version, and type of application such as Silverlight or XNA Framework. The following table lists and defines the attributes within the **Application** element:

Attribute	Definition
ProductID	Type: String. The default value is the GUID for the project (128 bit). During the application submission process, a new Product ID will be inserted into the manifest.
Title	Type: String. The default name of the project.
RuntimeType	Type: String. The entry will be either Silverlight or XNA Framework based on the application selection.
Version	Type: String. The default value is set to “1.0.0.0”.

Attribute	Definition
Genre	Type: String. The entry will be either “Apps.Normal” for Silverlight or “Apps.Game” for XNA Framework applications.
Author	Type: String. The project author’s name.
Description	Type: String. The text for a sample description.
Publisher	Type: String. This value defaults to empty in the file.

Icon Path: <IconPath></IconPath>

The **IconPath** element is a child element of the **Application** element and provides the location of the application icon that will be visible in the application list. The default image will be AppIconGeneric.png for Silverlight projects and GameThumbnail.png for XNA Framework games. The attributes are for internal use only.

Capabilities: <Capabilities></Capabilities>

The **Capabilities** element is a child element of the **Application** element. Windows Phone provides a capabilities-driven security model where a user must opt-in to certain functionality within the application. Some examples include using network-based services where a user could incur additional roaming costs if the use of the services were not disclosed in the application. Or, the use of push notifications that can also produce roaming charges. The primary goals of this capability model are to:

- Ensure proper disclosure – Users must be notified if an application’s functionality poses security risks. They must opt-in to allow the functionality to be activated.
- Decrease the attack surface - Capabilities are used to create a security chamber in which the application will execute. This chamber is created once at install-time and used from there-on for the application. The below tables highlights the current supported capabilities and their descriptions:

The **Capabilities** element has one attribute, **Name**, that is a string.

Capability ID/Name (Type: String)	Description
ID_CAP_NETWORKING	Applications with access to network services. This must be disclosed because services can incur cost when a phone is roaming.
ID_CAP_LOCATION	Applications with access to location services.
ID_CAP_SENSORS	Applications using the Windows Phone sensors.
ID_CAP_MICROPHONE	Applications that use the microphone. The application can record without visual indication that recording is taking place.
ID_CAP_MEDIALIB	Applications that can access the media library.
ID_CAP_GAMERSERVICES	Applications that can interact with Xbox LIVE APIs. This must be disclosed due to privacy issues since data is shared with Xbox.
ID_CAP_PHONEDIALER	Applications that can place phone calls. This may happen without visual indication for the end user.
ID_CAP_PUSH_NOTIFICATION	Applications that can receive push notifications from an Internet service. This must be disclosed as usage could incur roaming charges.
ID_CAP_WEBBROWSERCOMPONENT	Applications that use the web browser component. There are security risks with scripting.

When you are developing your application, if you do not need to use a specific capability, you can remove the entry. However, if you do not use a capability when needed, you will receive an **UnauthorizedAccessException** error with an “Access denied” message when you attempt to use the functionality in the application.

Tasks: <Tasks></Tasks>

The **Tasks** element is child element of the **Application** element. This element is for internal use only.

Attribute	Description
Name	Type: String. The default name will be “_default.”
NavigationPage	Type: String. This is the navigation page that a task will navigate to in the application when it starts. This is used by the Silverlight framework to target pages. It does not exist in the XNA Framework application manifest file.

Tiles: <Tokens></Tokens>

The **Tokens** element is a child element of the **Application** element. Each application is associated with a tile that a user can pin to the Quick Launch area of the Start experience. The sections below provide the names and describe the elements and attributes for tiles in the manifest.

Primary Tile: <Primary Token ></Primary Token >

The **Primary Token** element describes the tile that is associated with an application. It is a child element of the **Tokens** element.

Attribute	Description
Token ID	Type: String. The name of the tile that is set to "Default" by default.
TaskName	Type: String. The name of the task that the tile invokes on press.

Tile Template: < TemplateType5></ TemplateType5>

The **TemplateType5** element is the default template allowed for a tile. It is a child element of the **Tokens** element. This element also has the following nested child elements:

- BackgroundImageURI
- Count
- Title

Background Image URI: < BackgroundImageURI></ BackgroundImageURI>

The default tile image is ApplicationIcon.png for Silverlight, and GameThumbnail.png for XNA Framework.

Count: <Count></Count>

Type: Int. A tile can contain a counter element with values that range from a minimum of 0 to a maximum of 99. If the value is 0, nothing will appear on the tile.

Title: <Title></Title>

This is the title of the tile that appears on the start menu. The value defaults to the project name.

