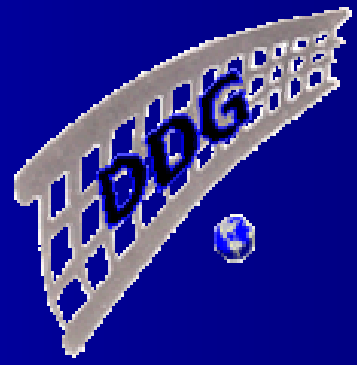
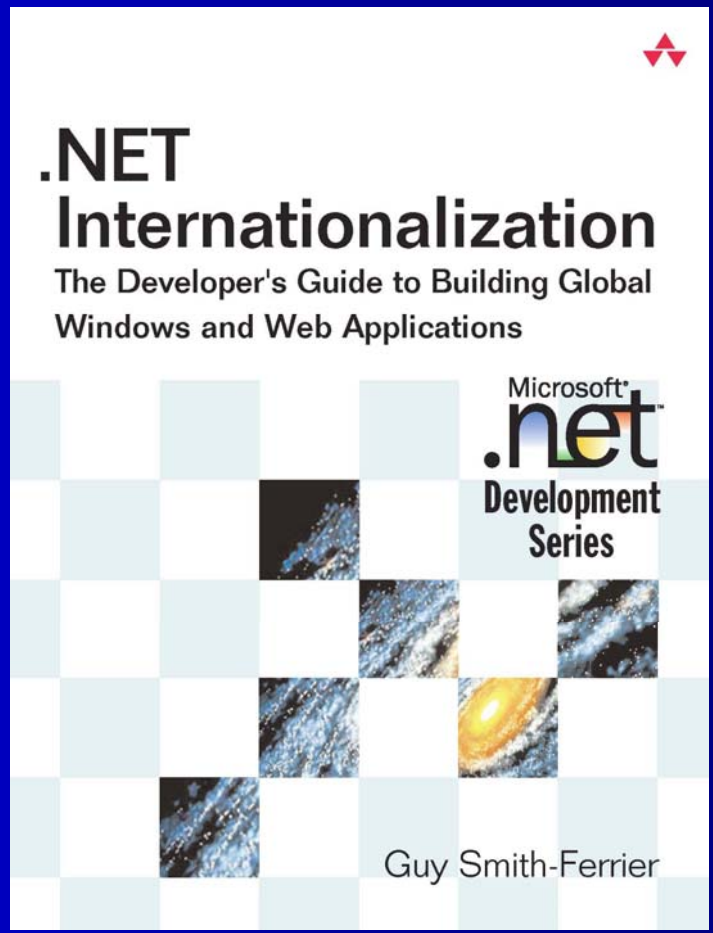


An Overview Of ClickOnce Deployment

Guy Smith-Ferrier
Courseware Online
gsmithferrier@coursewareonline.com

Author of…

- .NET Internationalization, Addison Wesley, ISBN 0321341384
  - Due Summer 2005



ClickOnce Vision

- To bring the ease and reliability of web application deployment to Windows Forms applications

Agenda

- Online applications
  - How it works
- Client Requirements
- Online and Offline applications
  - How it works
- The ClickOnce Cache
- ClickOnce Manifest Files
- Programmatic Updating
- Publishing Using msbuild
- ClickOnce Security
- ClickOnce vs. The Windows Installer

Information Sources

- Essential ClickOnce, Duncan Mackenzie, Addison Wesley
  - Not published yet
- The Magic Of ClickOnce
  - http://www.ondotnet.com/lpt/a/5205
- InstallSite: Microsoft ClickOnce Technology
  - http://www.installsite.org/pages/en/clickonce.htm
- ClickOnce In Visual Studio 2005
  - http://mtaulty.com/blog/archive/2004/07/05/524.aspx
- MSDN TV - Introducing ClickOnce: Web Deployment for Windows Forms Applications
  - http://msdn.microsoft.com/msdntv/episode.aspx?xml=episodes/en/20040108clickoncejc/manifest.xml

Information Sources (continued)

- Smart Client Developer Center Home: ClickOnce
  - http://msdn.microsoft.com/smartclient/understanding/windowsforms/2.0/features/clickonce.aspx
- Introducing Client Application Deployment with "ClickOnce"
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwinforms/html/clickonce.asp
- Deploying Windows Forms Applications with ClickOnce
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnreal/html/realworld12012004.asp
- Choosing Between ClickOnce and Windows Installer
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/clickoncevsinstaller.asp
- Deploy and Update Your Smart Client Projects Using a Central Server
  - http://msdn.microsoft.com/msdnmag/issues/04/05/clickonce/default.aspx

Information Sources (continued)

- ClickOnce Deployment for Windows Forms Applications
  - http://msdn2.microsoft.com/library/wh45kb66.aspx
- ClickOnce security and download on demand
  - http://blogs.msdn.com/misampso/archive/2004/05/06/127570.aspx
- How ClickOnce Manifest Generation works with MSBuild
  - http://www.windowsforms.net/articles/clickonce_msbuild.aspx
- ClickOnce and Security
  - http://www.develop.com/downloads/DM_ClickOnce.pdf
- Troubleshooting ClickOnce Deployment
  - http://msdn2.microsoft.com/library/ms165433(en-us,vs.80).aspx

Information Sources (continued)

- Configuring ClickOnce Trusted Publishers
  - http://msdn.microsoft.com/smartclient/default.aspx?pull=/library/en-us/dnwinforms/html/clickoncetrustpub.asp
- Simplify App Deployment with ClickOnce and Registration-Free COM
  - http://msdn.microsoft.com/msdnmag/issues/05/04/RegFreeCOM/default.aspx
- Automated Smart Client Deployment And Update
  - http://www.theserverside.net/articles/showarticle.tss?id=AutomatedSmartClient

Before ClickOnce

- .NET Framework 1.0 and 1.1 support Zero Touch Deployment (also called No Touch Deployment, HREF Deployment and HTTP Download)
- The Patterns And Practices Group released the Updater Application Block which supports some of the features of ClickOnce
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/updater.asp
- The Updater Application Block 2 is simpler and closer to ClickOnce
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag2/html/updaterv2.asp

Offline Example

- Create a Windows Forms application and call it CustomerCare1
- Set Form1.Text to "Customer Care" and add a button
- Select Build | Publish CustomerCare1
  - Click Next
  - Select the "No" radio button
  - Click Next
  - Click Finish
- In the CustomerCare1 web page click on the Run CustomerCare1 button
  - In the Security Warning dialog click on Run

Offline Example (continued)

- Close the application
- Click on the Run button again and observe that the application doesn't show the "Downloading" dialog and it starts up faster
- Close the application and close the web page
- Add a button to CustomerCare1, publish it again and click on the Run button
  - Observe that the application shows the "Downloading" dialog as it downloads the updated copy

How Does It Work ?

- The Publish Wizard:-
  - creates a new Virtual Directory in IIS using the name of the application (e.g. CustomerCare1)
  - creates a new folder beneath the Home Directory (e.g. C:\Inetpub\wwwroot\CustomerCare1)
    - adds a customized setup.exe (the Bootstrapper) to the new folder
    - adds a Publish.htm page which is shown to the user
    - adds a deployment manifest (e.g. CustomerCare1.application)
    - adds a version-specific deployment manifest (e.g. CustomerCare1_1_0_0_0.application)
  - creates a new folder for the latest version of the application (e.g. CustomerCare1_1.0.0.0)
    - adds the application (e.g. CustomerCare1.exe.deploy)
    - adds an application manifest (e.g. CustomerCare1.exe.manifest)
- When the application is run it is cached so it runs faster next time

How Does Publish.htm Work ?

- Publish.htm includes:-
  - a Button called InstallButton with an href of "setup.exe"
  - a section called BootstrapperSection
- Publish.htm includes JavaScript which is automatically executed to determine if the .NET Framework 2.0 is installed
  - If it is installed:-
    - BootstrapperSection.style.display is set to none
    - InstallButton.href is set to the application's deployment manifest (e.g. CustomerCare1.application)

ClickOnce Client Requirements

- For browser access the client must support URL activation e.g.
  - Internet Explorer
    - Non-Internet Explorer browsers (e.g. Mozilla, Firefox) will not work because they look for the application files in the same location as the manifest files
  - Outlook
  - MSN Explorer
  - AOL

ClickOnce Client Requirements (continued)

- Before the application can be run the client must have:-
  - Windows 98 or higher
  - the .NET Framework 2.0
  - The Windows Installer 2.0
  - Whatever pre-requisites have been dictated by the publisher
- ClickOnce can install all of these using the ClickOnce Bootstrapper
  - The Bootstrapper requires Administrator access to install pre-requisites (but not to install the ClickOnce application)

Online And Offline Example

- Create a Windows Forms application and call it CustomerCare2
- Set Form1.Text to "Customer Care 2" and add a button
- Select Build | Publish CustomerCare2
  - Click Next
  - Select the "Yes" radio button
  - Click Next
  - Click Finish
- In the CustomerCare2 web page click on the Install button
  - In the Security Warning dialog click on Install

Online And Offline Example (continued)

- Show that an item has been added to the Start menu
- Show that an item has been added to Add/Remove Programs
- Close the application and close the web page
- Add a button to the form and publish it again
- In Internet Information Services select the Default Web Site, right click and select Stop
- In Windows select Start | All Programs | <Company Name> | CustomerCare2
  - The original, 'cached' program is run and not the new version

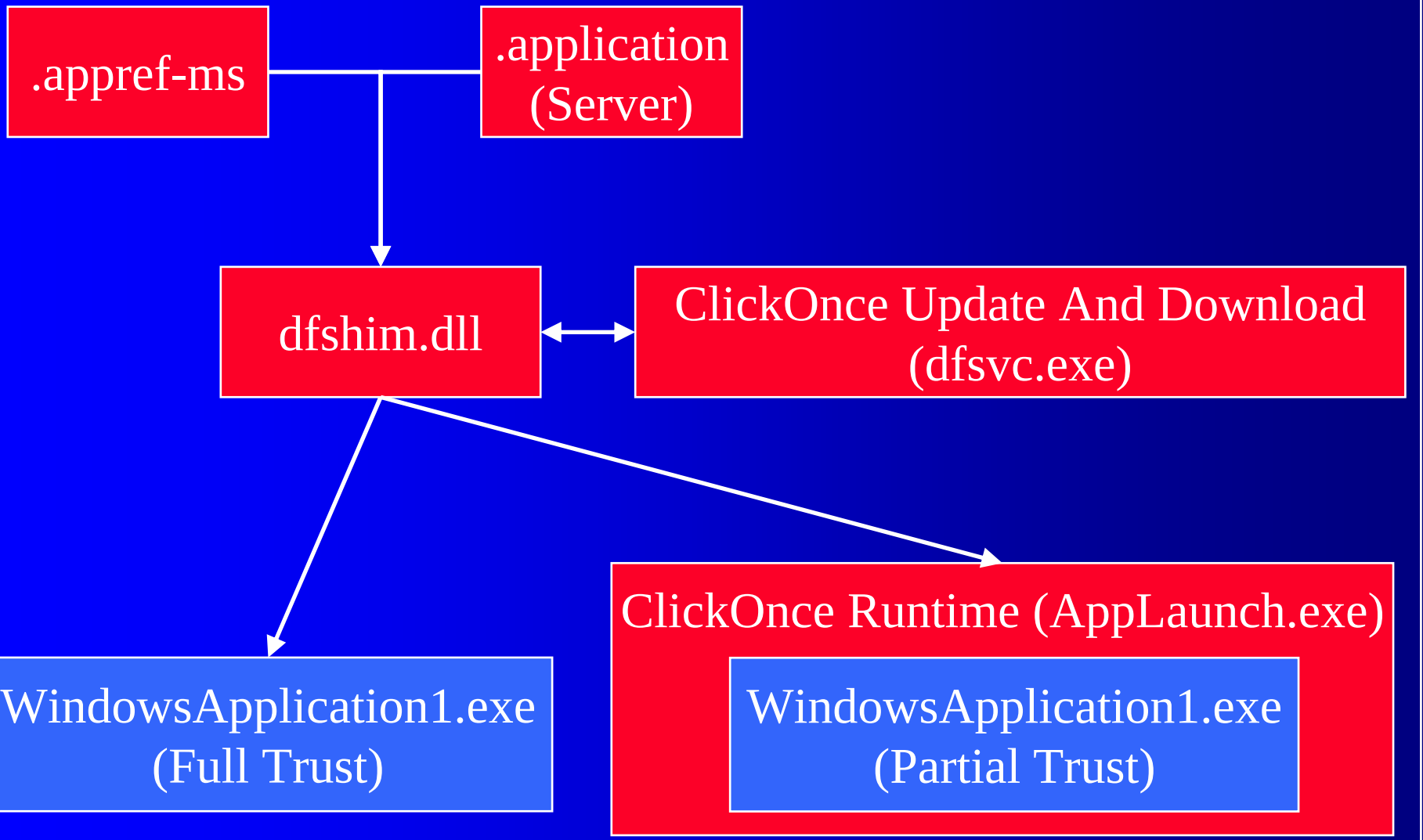
Online And Offline Example (continued)

- Close the application
- Restart the Default Web Site
- Run the application
  - In the Update Available dialog click Ok and the updated application is run

How Does It Work ?

- The Publish Wizard performs the same steps as before except:-
  - Publish.htm says "Install" instead of "Run" (but the functionality is identical)
- The shortcut added to the Start menu is a .appref-ms file
  - .appref-ms files are MIME files and are automatically executed by the operating system's MIME filter
  - The .appref-ms file contains the URL to the .application file (the deployment manifest) on the web server
  - .application and .appref-ms extensions invoke COM routines in dfshim.dll which launch the ClickOnce update and download component
  - The ClickOnce update and download component (dfsvc.exe) manages the deployment and update of the application
    - dfsvc.exe allegedly times out after 15 minutes of inactivity
  - If the application is partially trusted the ClickOnce runtime (AppLaunch.exe) is run to handle the sandboxing and execution of the application
  - Otherwise the application is run normally

How Does It Work ? (continued)



The ClickOnce Cache

- Online and installed applications are held on the client computer in the ClickOnce cache
- The cache is in <Documents>\<User>\Local Settings\Apps e.g.
  `\Documents and Settings\Administrator\Local Settings\Apps`
- The cache contains the complete application
  - The application's assemblies, its configuration files, its data
- Applications cannot be installed outside of the cache
  - Applications cannot be installed to a specific location
- There is no publicly available "ClickOnce Cache API"
  - The cache can be managed by regular file I/O methods

The ClickOnce Cache Size

- The maximum size for all online applications is 100Mb
  - This can be changed in the registry:-
  - HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Deployment\OnlineAppQuotaInKB
  - The value is in Kb so 200Mb is 204800
- There is no restriction on installed applications
  - Installed applications do not count towards the online applications maximum size
- The cache contains the current and previous version:-
  - Cached online applications are aged
    - When the cache is full the least recently used application(s) is/are deleted
      - Old online apps can be programmatically purged using file delete methods
  - Cached installed applications are not aged

ClickOnce Manifest Files

	Deployment Manifest	Application Manifest
Description	Contains information about the application including the current version number	Contains information about the files, dependencies and security requirements of a specific version
Responsibility	Administrator	Developer
Extension	.application	.exe.manifest
Location	The published folder e.g. C:\Inetpub\wwwroot\WindowsApplication1	The specific version of the published application e.g. \Inetpub\wwwroot\WindowsApplication1\ WindowsApplication1_1.0.0.3
Signing	Deployment manifest files are always signed (using XMLDSIG) and cannot be tampered with	Application manifest files are always signed (using XMLDSIG) and cannot be tampered with

Manifest Generation and Editing Tool (MAGE)

- The Manifest Generation and Editing Tool (MAGE) is a .NET Framework 2.0 SDK tool for creating and editing ClickOnce manifest files
  - There are supposed to be two versions (Mage.exe is a command line tool and MageUI.exe is a GUI) but they both do the same thing in Beta 2
- MAGE and Visual Studio 2005 should offer similar functionality by the time that Visual Studio 2005 ships

Command Line Parameters

- ClickOnce applications can accept command line parameters as arguments in the application manifest's URL

```
http://localhost/WindowsApplication1.application?Culture=en-GB
```

- Edit the .appref-ms file (in \Documents and Settings\<UserName>\Start Menu\Programs\<CompanyName>\WindowsApplication1) using NotePad
- Command line parameters must be enabled
  - In the project's Publish properties click the Options… button and check "Allow URL parameters to be passed to application"

Programmatic Updating

- Turn off automatic updating
  - In the Publish properties, click the Updates… button and uncheck the "Application should check for updates" checkbox
- Add a reference to System.Deployment.dll
- Add the following directive to the code:-

```
using System.Deployment.Application;
```

- Add a button to the main form with the following code:-

Programmatic Updating (continued)

```
ApplicationDeployment applicationDeployment =
   ApplicationDeployment.CurrentDeployment;
if (applicationDeployment.CheckForUpdate())
{
    applicationDeployment.Update();
    Application.Restart();
}
```

- Publish the application and install it
- Change the application and publish it again
- Run the application and show that the new version is not automatically downloaded
- Click on the button to update the application

Programmatic Updating (continued)

- Change the button's code to:-

```
ApplicationDeployment deployment =
    ApplicationDeployment.CurrentDeployment;
UpdateCheckInfo updateCheckInfo = deployment.CheckForDetailedUpdate();
if (updateCheckInfo.UpdateAvailable)
{
    if (MessageBox.Show("An update is available" +
        System.Environment.NewLine + "Version " +
        updateCheckInfo.AvailableVersion + " (" +
        updateCheckInfo.UpdateSizeBytes.ToString() + " bytes)" +
        System.Environment.NewLine +
        "Update now ?", "", MessageBoxButtons.YesNo) ==
        DialogResult.Yes)
    {
        deployment.Update();
        Application.Restart();
    }
}
```

Uses For Programmatic Updating

- Updates on demand
  - The user can check for updates manually
- Download load balancing
  - The client could contain logic which determines which day or hour the user checks for downloads based upon data specific to the client (e.g. IP address, first letter of user name)
- Beta programs
  - The client could compare the new minor version number with the old and only download it if the client has been registered as a beta client

Publishing A ClickOnce Application Using msbuild

- To publish a ClickOnce application without using Visual Studio 2005 use the .NET Framework 2.0 SDK's msbuild

```
msbuild /target:publish
```

- Not all IDE settings apply to msbuild
  - "Automatically increment revision with each release" is not respected when using msbuild
- msbuild does not copy the published application from the "publish" folder to the deployment location
  - You must perform this step manually
- To publish to a specific location use the PublishUrl property:-

```
msbuild /target:publish /property:PublishUrl=
http://localhost/WindowsApplication1
```

ClickOnce And Internationalized Applications

- If you publish a Windows Forms application which has satellite assemblies the satellite assemblies will not be included in the manifest by default
- Set the "Publish language" in Visual Studio 2005 to the language which should be deployed with the application
  - The "Publish language" also affects the language used in ClickOnce dialogs
    - You need to install the .NET Framework Language Pack which matches the "Publish language" if you want your ClickOnce dialogs to use that language

Publishing ClickOnce Internationalized Applications Using msbuild

- To publish for a single culture:-

```
msbuild /target:publish /property:TargetCulture=fr
```

- To publish for all cultures:-

```
msbuild /target:publish /property:TargetCulture=*
```

- In addition you might want to:-
  - Specify a localized Product Name

```
msbuild /target:publish /property:ProductName="Zona De Tiempo"
```

  - Add one or more .NET Framework Language Packs to the pre-requisites

Manifest Signing

- Application and deployment manifests are signed using an XML Digital Signature (XMLDSIG)
  - Manifests which are altered without being signed again result in the error:-

```
The manifest XML signature is invalid
```

- Assemblies are signed using the publisher certificate (.pfx)
  - You cannot sign assemblies using a strong name (.snk)
  - Signing assemblies with a strong name would be bad in a ClickOnce application:-
    - An assembly which uses a signed assembly must be rebuilt to use the new signed assembly when the signed assembly is changed
      - Changing one assembly could mean rebuilding many assemblies and therefore cause unnecessary downloads

ClickOnce Security Zones

- ClickOnce applications adopt the security zone that they are installed from
  - Not the location they are run from

ClickOnce Application	Security Zone
Online only	Internet
Online and offline installed from web	Internet
Online and offline installed from file server	Local Intranet
Offline installed from CD-ROM	Local Computer (Full Trust)

Calculating Security Permissions

- You can calculate your applications' security permissions in two ways:-
  - In Visual Studio open the Security tab of the project's Properties
    - Click on the Calculate Permissions… button
  - In a command prompt run permcalc.exe:-

```
permcalc WindowsApplication1.exe
```

Debug In Zone

- If you build, publish and debug your application on a single machine your debugging experience will be unrealistic
  - You will be debugging with Full Trust
- Visual Studio can set the permissions which are used during debugging to a set of permissions other than full trust
  - In Solution Explorer right click the project and select Properties
    - Select the Security tab and click "Enable ClickOnce Security Settings"
    - In the "Zone your application will be installed from" combo box set the settings to either Local Intranet or Internet

Trust Licenses

- Trust licenses allow you to create ClickOnce applications which are trusted without the user being involved with the trust process
  - The benefit is that you can deploy ClickOnce applications which have elevated security without your users having to answer trust questions

The Trusted Application Deployment Process

- The user's machine is configured to accept trust licenses from the given trust license issuer
  - This is a one-time touch to the user's machine using the .NET 2.0 Configuration Tool
- The developer requests a trust license (.tlic file) from the trust issuer and gives the issuer the application manifest (which includes the requested permissions)
- The trust issuer returns a trust license to the developer
- The developer signs the application with the trust license

ClickOnce vs The Windows Installer

	ClickOnce	The Windows Installer
Specify The Installation Folder	No	Yes
Install For All Users	No	Yes
Create File Associations	No	Yes
Create Shell Extensions	No	Yes
Install .NET Components In The GAC	No	Yes
Install COM Components In The Registry	No	Yes
Run Custom Actions	No	Yes
Write To Registry	No	Yes
Administer ODBC	No	Yes
Create And Apply .msp Patches	No	Yes

ClickOnce And COM Components

- ClickOnce applications can use COM components. You have three choices:-
  - If your clients use Windows XP then Visual Studio can generate an application manifest which uses Registry-Free COM components
  - If you can't use Registry-Free COM you can install the COM components as a custom pre-requisite using the ClickOnce Bootstrapper
    - The COM components will not be part of the application manifest and will not be auto-updated
  - If you can't use Registry-Free COM and you want the COM components to be auto-updated add them to the application and change the application so that it invokes the COM object entry points directly (without using COM)

ClickOnce And Longhorn

- The following enhancements are expected in the Longhorn version of ClickOnce:-
  - Background Intelligent Transfer Service (BITS)
  - Create File Associations (without requesting additional security permissions)
  - Specify the application's privacy settings
  - Create Shell Extensions

ClickOnce And .NET Framework 1.1 Windows Forms Applications

- You can use ClickOnce to install .NET Framework 1.1 Windows Forms Applications
  - ClickOnce requires that the entry point into the application is a .NET Framework 2.0 application
    - Your .NET Framework 2.0 application can simply invoke your .NET Framework 1.1 application
      - Invoking a process requires a higher level of trust
  - Add a custom prerequisite for the .NET Framework 1.1

Summary

- ClickOnce brings the ease of web deployment and update to Windows Forms applications
- The ClickOnce runtime and ClickOnce cache ensure there is minimal touch to the client
- ClickOnce applications have many options for configuring deployment and updates including programmatic updates
- Security is build in to ClickOnce and is not optional