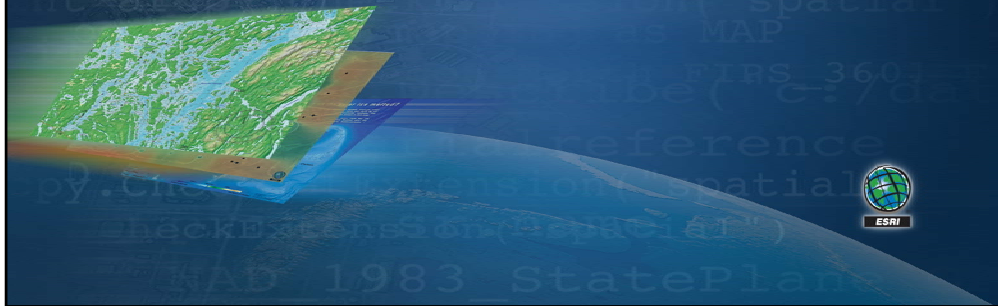


# ESRI Developer Summit

March 22–25, 2010  
Palm Springs, CA

## Introducing Desktop Add-Ins

*Russell Louks*  
*Steve Van Esch*



## ArcGIS Desktop Add-Ins

*A new way to customize and extend  
ArcGIS Desktop applications.*

- Easier to build
- Easy to share
- More secure
- Pluggable Architecture



ESRI provides project templates/wizards for Visual Studio, Visual Studio Express Editions, and Eclipse. Programming tasks associated with COM registration and component categories are unnecessary with Add-Ins. The declarative programming model makes for less methods / properties to implement in code. This model also lets the framework optimize certain common scenarios; for instance, commands/tools can be displayed without actually loading their associated assemblies/DLLs.

An Add-In consists of a single file, which can be “installed” by simply double clicking the file in Explorer, in a Mail client, or in a Web browser. Add-Ins require no installation program, no registration step and do not require administrative rights to get them working. Add-In files can be managed on network shares where they can be updated even if in use, clients will pickup the updated version the next time they start the associated application.

Add-In files can be digitally signed so that users can determine the source of the file, and further determine if the file has been tampered with or altered in any way since it was published. Users can configure ArcGIS applications to reject any Add-In file which isn’t signed with a certificate issued by a trusted authority.

The Add-In framework can be extended to cover additional language types. Currently .ESRI supports C#, VB.NET, and Java; ESRI currently evaluating Python and C++ for a subsequent release.

## **Add-In Types supported at ArcGIS 10.0**

- Buttons
- Tools
- Combo Boxes
- Multi-Items
- Menus
- Context Menus
- Toolbars
- Tool Palettes
- Dockable Windows
- Application Extensions
- Editor Extensions

Add-Ins support the most common extensibility types.  
New types will be added in subsequent releases.

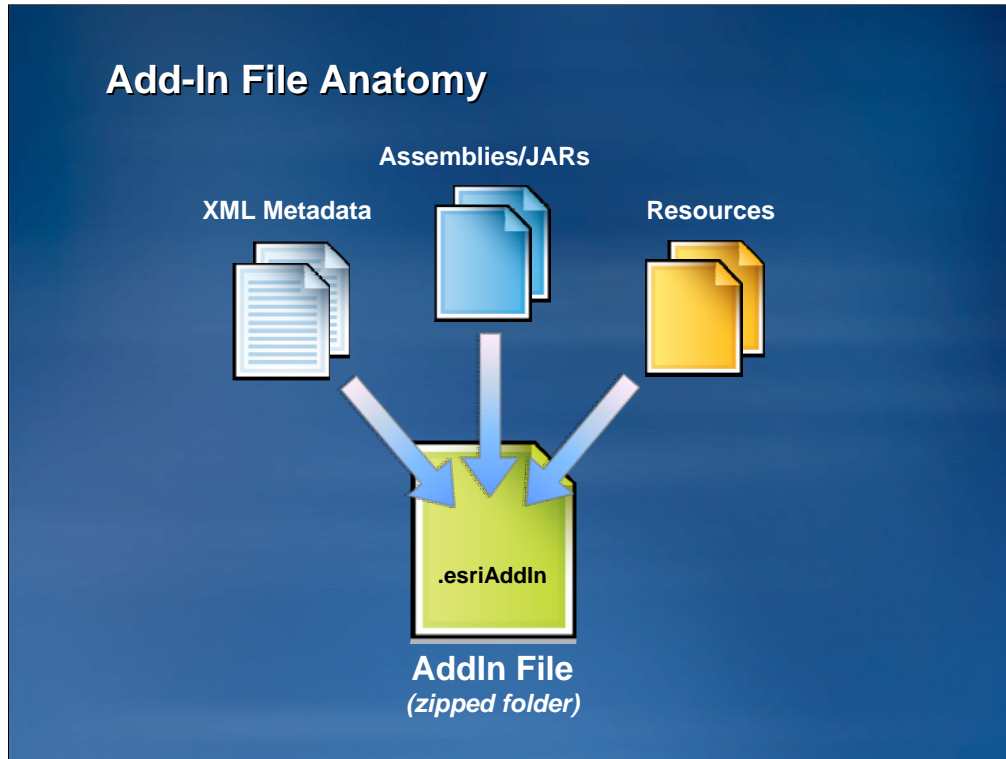
## Supported Development Environments for Add-Ins at ArcGIS 10.0

- Visual Studio 2008 + .Net 3.5
- Visual Studio 2010 + .Net 3.5 or .Net 4.0
- Visual Basic 2008 Express Edition + .Net 3.5
- Visual C# 2008 Express Edition + .Net 3.5
- Visual Basic 2010 Express Edition + .Net 3.5 or .Net 4.0
- Visual C# 2008 Express Edition + .Net 3.5 or .Net 4.0
- Eclipse IDE for Java Developers

Express editions of Visual C# and Visual Basic, and Eclipse are freely downloadable.

You must install the appropriate SDK to get the Add-In project wizard/templates for your development environment. The .NET and Java ArcObjects SDKs are shipped with the desktop products.

Support for VS2010 and .NET 4.0 is not yet available in the 10.0 pre-release pending the release of VS2010. Support will be included in the release of ArcGIS 10.0



All of the necessary components that make up an Add-In are combined into a single compressed archive (zipped folder). If you rename an Add-In file's extension to zip, you can open the file in explorer and examine the contents.

The three primary components of any Add-In file are:

#### **The declarative aspect**

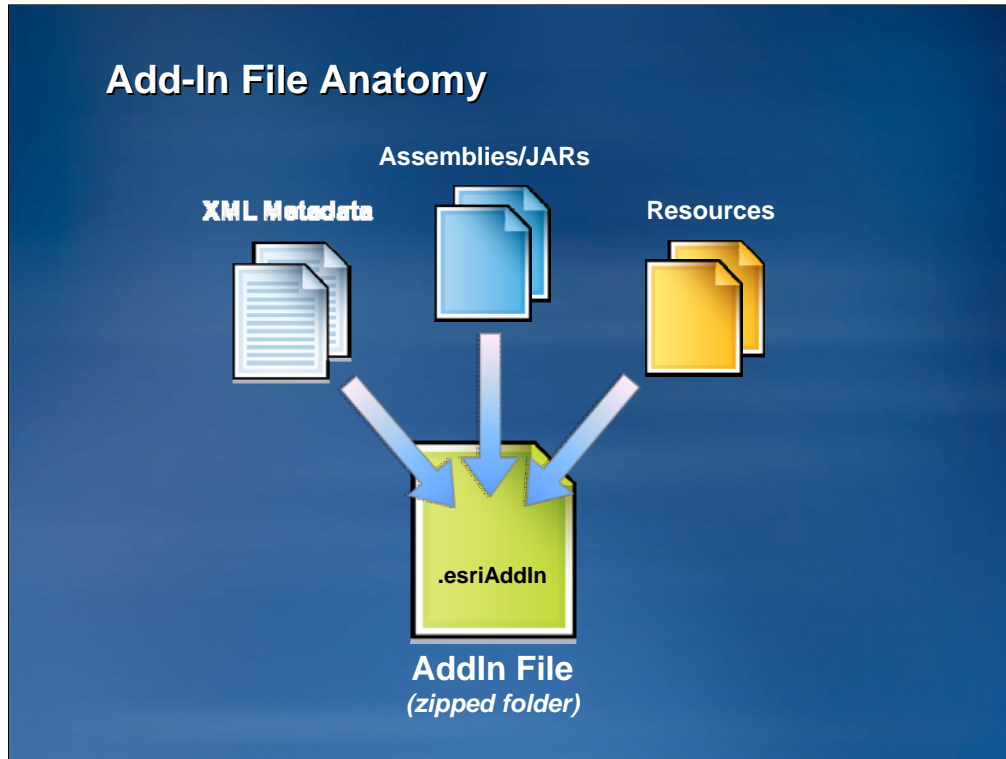
XML metadata that describes static aspects of the Add-In including such things as captions, categories, tooltips, and other configuration information.

#### **•The programmatic aspect**

The assemblies, JAR files, etc. that provide the logic and behavior of the Add-In.

#### **•Resources**

Supporting resources including images referenced by the XML, debugging symbol files, localized resource files, or internal data.



All of the necessary components that make up an Add-In are combined into a single compressed archive (zipped folder). If you rename an Add-In file's extension to zip, you can open the file in explorer and examine the contents.

The three primary components of any Add-In file are:

**The declarative aspect**

XML metadata that describes static aspects of the Add-In including such things as captions, categories, tooltips, and other configuration information.

**•The programmatic aspect**

The assemblies, JAR files, etc. that provide the logic and behavior of the Add-In.

**•Resources**

Supporting resources including images referenced by the XML, debugging symbol files, localized resource files, or internal data.

## Anatomy

### Metadata (declarative aspect)

- XML describes all the static aspects of the add-in
  - ID
  - Target
  - Author
  - Version
  - Company
  - Website
  - Description
  - Caption
  - Tooltip
  - Help
  - Image
  - Category
  - Toolbars / Menus
  - Docking state / Position
  - ...

### Config.xml

```
<ESRI.Configuration
  xmlns="http://schemas.esri.com/Desktop/AddIns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Name>Acme Custom Mapping Extension</Name>
  <AddInID>{69b202bd-6191-4674-a9ee-e4bab0522828}</AddInID>
  <Description>Acme Co. custom mapping extension for
    advanced parcel editing.</Description>
  <Version>1.0</Version>
  <Image>Images\Acme.png</Image>
  <Author>John Locke</Author>
  <Company>Acme Co.</Company>
  <Date>3/15/2010</Date>
  <Targets>
    <Target name="Desktop" version="10.0" />
  </Targets>
  <AddIn language="CLR" library="Acme.dll" namespace="Acme">
    <ArcMap>
      <Extensions>
        <Extension id="Acme_ParcelEditor_Extension"
          class="MainExt" />
      </Extensions>
      <Commands>
        <Button id="Acme_ParcelEditor_ToggleDockWin"
          class="ToggleDockWin"
          caption="Open Window"
          category="Acme Tools"
          image="Images\ToggleDockWin.png"
          tip="Open Parcel Window"
          message="Opens the parcel editing window.">
        </Button>
      </Commands>
      <Toolbars>
        <Toolbar id="Acme_ParcelEditor_Toolbar"
          caption="Acme Parcel Editor">
          <Items>
            <Button refID="Acme_ParcelEditor_ToggleDockWin"/>
          </Items>
        </Toolbar>
      </Toolbars>
    </ArcMap>
  </AddIn>
</ESRI.Configuration>
```

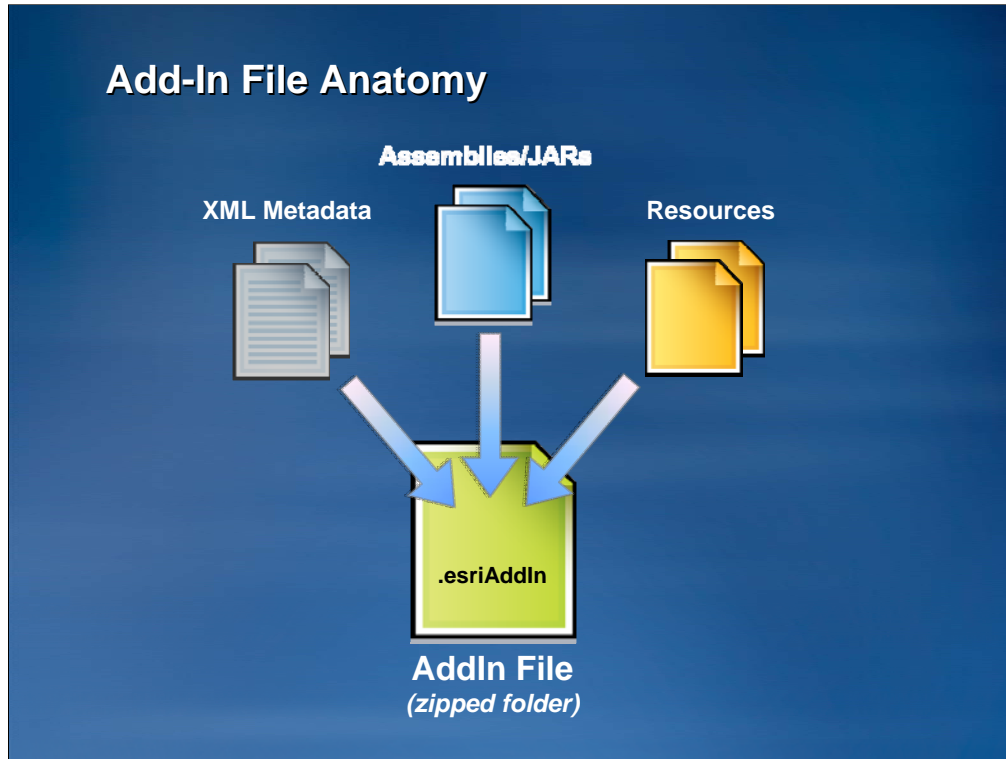
Many aspects which have traditionally been supplied through methods are handled declaratively in the Add-In model using XML. Items such as Captions, Tooltips, Help, etc., can be queried for by the system without actually creating the component. This makes the system more efficient while reducing the amount of code which needs to be written. Certain kinds of simple behavior can also be specified declarative; for example, whether to load extensions on demand or at startup, where dockable windows should appear, etc. Menus and toolbars are now entirely declarative.

Note that Add-In components are identified using ordinary strings rather than GUIDs. These IDs should still be unique to avoid ID collisions and should be named using a namespace convention:

<company name>\_<assembly>\_<ID>

The project templates will automatically generate a unique GUID which identifies the Add-In file itself. This is done to prevent file name conflicts within the well known folders.





All of the necessary components that make up an Add-In are combined into a single compressed archive (zipped folder). If you rename an Add-In file's extension to zip, you can open the file in explorer and examine the contents.

The three primary components of any Add-In file are:

**The declarative aspect**

XML metadata that describes static aspects of the Add-In including such things as captions, categories, tooltips, and other configuration information.

**•The programmatic aspect**

The assemblies, JAR files, etc. that provide the logic and behavior of the Add-In.

**•Resources**

Supporting resources including images referenced by the XML, debugging symbol files, localized resource files, or internal data.



## Anatomy

### Programmatic Aspect

- Add-In Behavior coded using
  - Visual Studio / Eclipse wizards and templates
  - Base classes for each Add-In type
  - Full ArcObjects API + programming environment (.NET/Java)

```
public class Button1 : ESRI.ArcGIS.Desktop.AddIns.Button
{
    protected override void OnClick()
    {
        ArcMap.Application.Caption = "Hello World!";
    }
}
```

An Add-In's programmatic or behavioral aspect is provided by implementing classes in .NET or Java. Base classes are provided to simplify this task; note that very few methods/properties are needed since many aspects are already covered by the declarative aspect.

Some example differences from classic COM base classes:

- The OnCreate method does not exist; initialization should be performed within the constructor instead.
- Access to the rest of the application is provided by the wizard generate static app class (e.g. ArcMap):

```
ArcMap.Application.Caption = "A new caption";
```

## Anatomy - Classic (managed) COM Button comparison

```
namespace ArcMapClassLibrary1
{
    /// <summary>
    /// Summary description for Command1.
    /// </summary>
    [Guid("dfab35ef-96f7-456b-9162-42d36f7a9e57")]
    [ClassInterface(ClassInterfaceType.None)]
    [ProgId("ArcMapClassLibrary1.Command1")]
    public sealed class Command1 : BaseCommand
    {
        #region COM Registration Function(s)
        [ComRegisterFunction()]
        [ComVisible(false)]
        static void RegisterFunction(Type registerType)
        {
            // Required for ArcGIS Component Category Registrar support
            ArcGISCategoryRegistration(registerType);

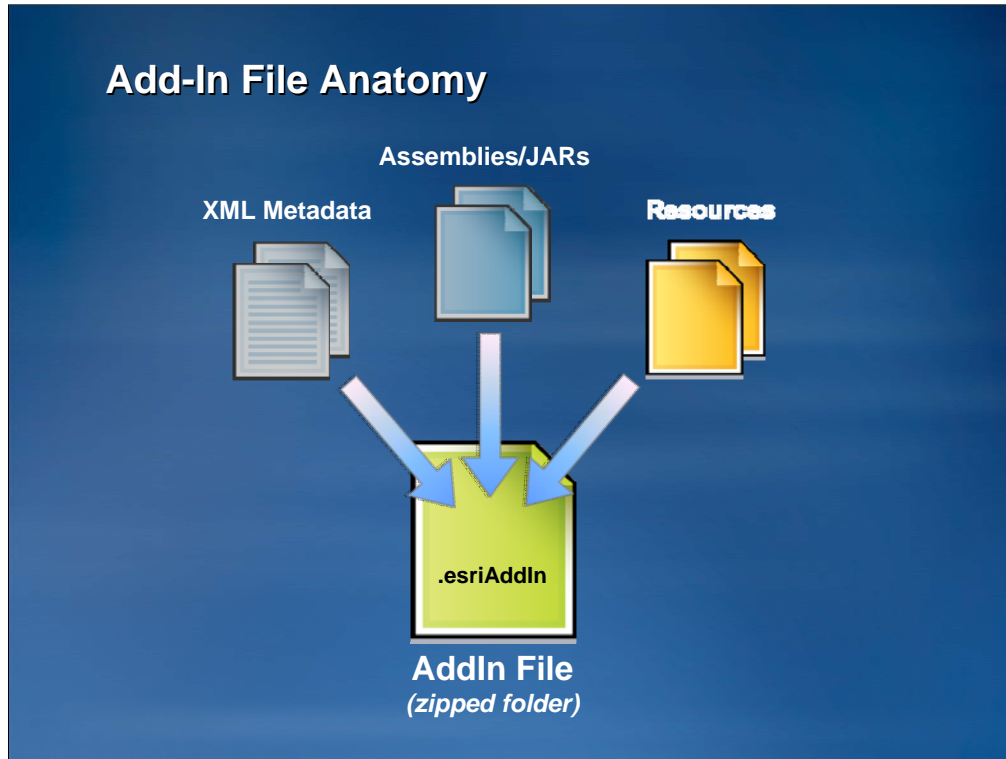
            //
            // TODO: Add any COM registration code here
            //
        }

        [ComUnregisterFunction()]
        [ComVisible(false)]
        static void UnregisterFunction(Type registerType)
        {
            // Required for ArcGIS Component Category Registrar support
            ArcGISCategoryUnregistration(registerType);

            //
            // TODO: Add any COM unregistration code here
            //
        }

        #region ArcGIS Component Category Registrar generated code
        /// <summary>
        /// Required method for ArcGIS Component Category registration -
        /// Do not modify the contents of this method with the code editor.
        /// </summary>
        private static void ArcGISCategoryRegistration(Type registerType)
```

The support logic needed to create an Add-In component is significantly smaller than the equivalent in the classic COM version (shown above).



All of the necessary components that make up an Add-In are combined into a single compressed archive (zipped folder). If you rename an Add-In file's extension to zip, you can open the file in explorer and examine the contents.

The three primary components of any Add-In file are:

**The declarative aspect**

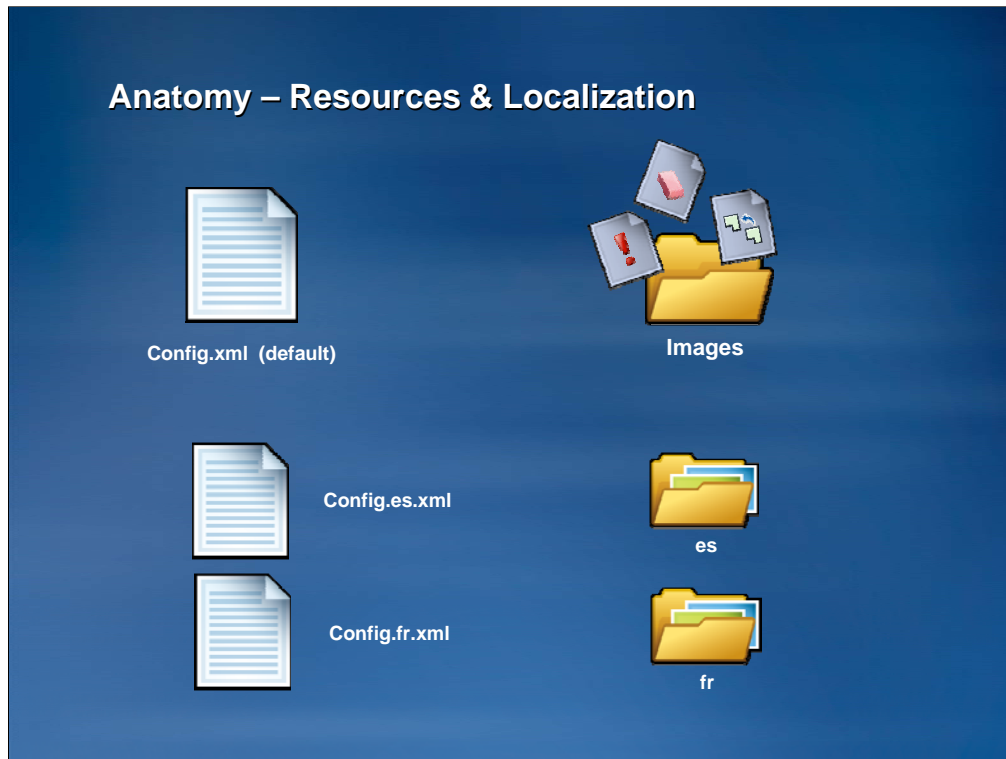
XML metadata that describes static aspects of the Add-In including such things as captions, categories, tooltips, and other configuration information.

**•The programmatic aspect**

The assemblies, JAR files, etc. that provide the logic and behavior of the Add-In.

**•Resources**

Supporting resources including images referenced by the XML, debugging symbol files, localized resource files, or internal data.



Using the standard local naming convention, localized versions of the config xml file can be established in the root of the zipped .esriAddIn. The appropriate version will automatically be loaded by the application depending on the user's locale and system settings. Developers can use this to create a single Add-In file which will work in several different locales/languages.

Resources may be added to the assembly itself, or separately by adding the required file/s to the project and setting "Copy to Output" to true; this will cause them to be incorporated into the compressed .esriAddin file. Resources added in this way can be accessed programmatically by the Add-In logic using the executing assembly's path.

GUI Forms and text within the language specific assemblies must be localized independently using the development environment's recommend procedure.

## Add-In File Discovery & Sharing



- Add-In files automatically discovered in well known local folders and incorporated into the Desktop applications at runtime.
- Folders are per user and per ArcGIS version

Under Windows Vista & Windows 7:

`C:\Users\<username>\Documents\ArcGIS\AddIns\Desktop10.0`

Under Windows XP:

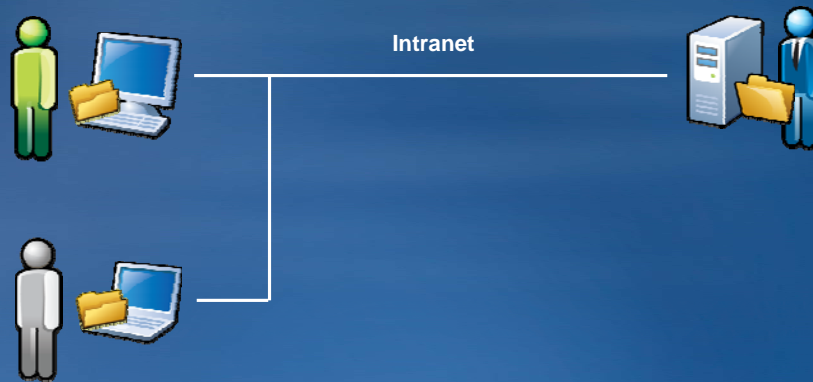
`C:\Documents and Settings\<username>\My Documents\ArcGIS\AddIns\Desktop10.0`

Add-In files must be placed in a specific well known folder under the user's account. Once an Add-In is built by a developer, it can be distributed to other users via email, published links, thumb drives, etc. The file then only needs to be copied to the appropriate well known folder within the recipient's machine. Add-In files can also be uploaded to websites such as ArcGIS Online.

Well known folders are named for the version of ArcGIS the Add-In was created for.

## Add-In File Discovery & Sharing

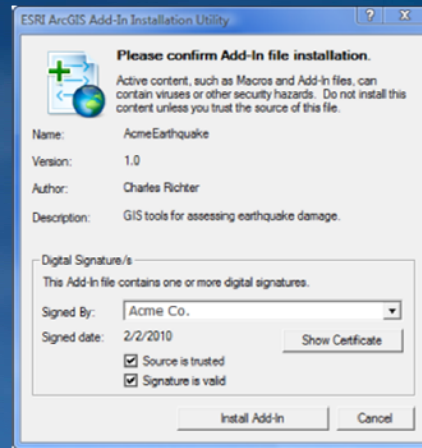
- Administered network shares
- Simplifies Updates



Alternative, an enterprise file share can be established to simplify frequent updates /bug fixes to the Add-In components. Users only need to add the appropriate share into their Add-In search path within the Add-In manager dialog. Newer versions of Add-In files discovered on these shares will be automatically copied to the local folders on the client machines. These updates can occur even if the Add-In file is presently in use; applications will get the latest version the next time they are restarted. Note that these shares should be read only for all non-administrators.

## Managing Add-Ins - Installation Utility

- Double-click “Install”
- Customize - Add From File
- XCOPY



Though Add-Ins can be manually copied to the appropriate well known folder, ESRI provides a safer alternative in the form of the Add-In Installation utility. This utility is associated with the standard Add-In file extension and will be automatically launched when a user double clicks on a Add-In file name within Explorer, an Email attachment, or a web link. Using the utility has several significant advantages over using XCOPY:

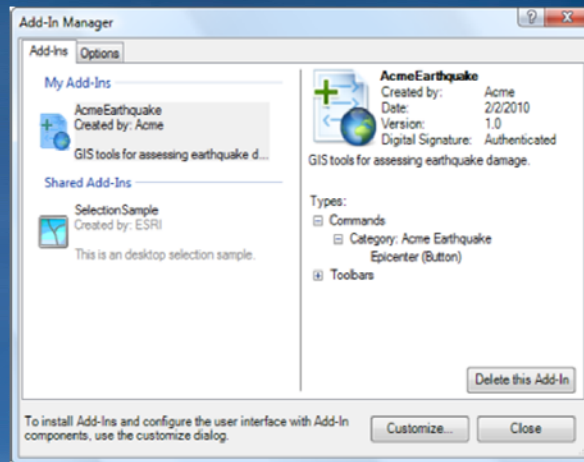
- 1.) The user doesn't have to browse for or be concerned with the Destination well known folder as the utility will read the metadata And copy the file to the appropriate location.
- 2.) The version of the Add-In file is checked against any existing version, so that a newer version is never overwritten by an older version.
- 3.) The user has an opportunity to review what's contained within the Add-In including security information.

Add-Ins can also be installed using the Customization Dialog's Add From File option.



## Managing Add-Ins – Add-In Manager Dialog

- Shows detailed information on all installed Add-Ins
- Mine vs. Shared
- Deleting Add-Ins



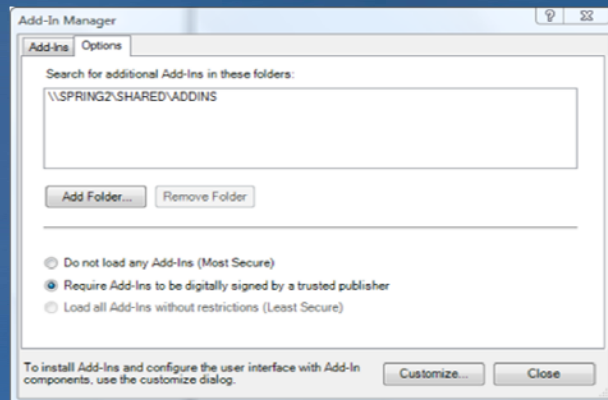
The Add-In manager dialog –available from the Customize menu within any of the ArcGIS desktop applications—displays detailed information on all the Add-Ins presently installed on a machine. In addition basic metadata and security information, users can also review a list of all the components contained within the Add-In file.

Add-In files are listed under two categories: My Add-Ins, and Shared Add-Ins. Shared Add-Ins are those that are references from network shares. My Add-Ins are those installed on the local machine. Only local Add-Ins can be deleted using the “Delete Add-In file” button.

Customizing the user interface with Add-In components such as buttons and tools, is accomplished using the customize dialog. A button is provided as a shortcut to the customize dialog from the Add-In manager dialog.

## Managing Add-Ins – Sharing & Security

- Custom search folders
- Security Settings



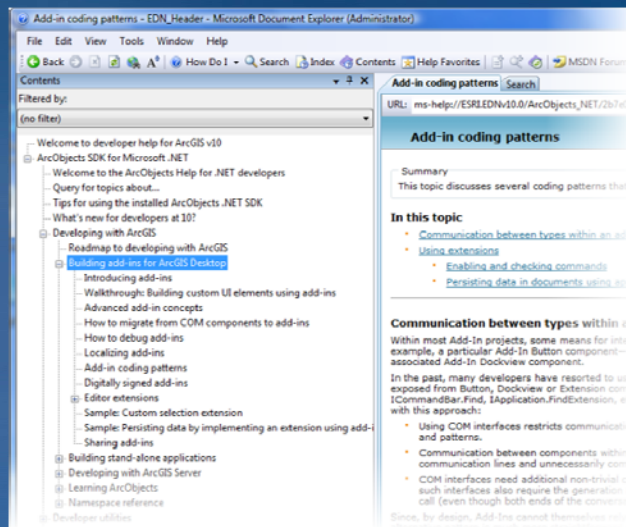
The options tab on the Add-In manager dialog provides access to control of shared folders and security. The system can be configured to use one of three security levels: the first disabled Add-Ins entirely, the second will allow only trusted and valid Add-Ins (those with digital signatures) to be loaded, the last setting (default) allows any Installed Add-In to load (including those without digital signatures).

Administrators can configure these settings for all users. Non-admin users can also change these settings, but can never lower the security level below what has already been established the an administrator.

## **Managing Add-Ins**

### **Installation and Sharing Demonstration**

## Authoring Add-Ins with .NET – Getting Help



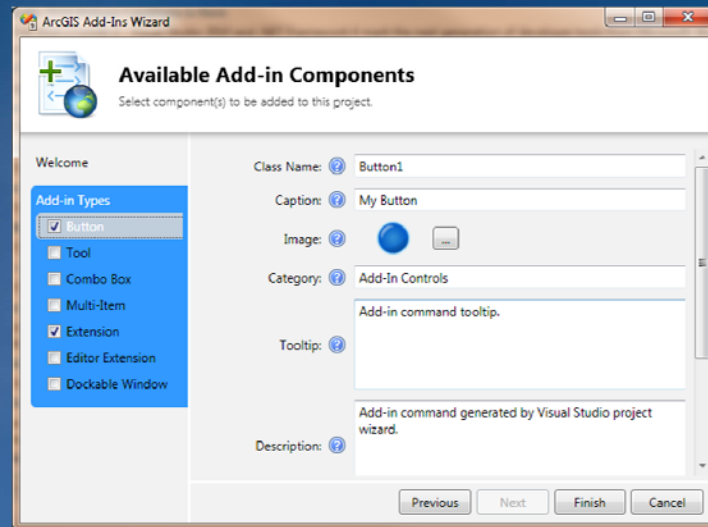
Developer Help – Building Add-Ins for ArcGIS Desktop

Detailed help has been created to help you through the process of understanding and authoring Add-Ins.

Add-In related information is located within the Developer Help under Developing with ArcGIS, Building Add-Ins for ArcGIS Desktop.

Topics include Development Walkthroughs, Migrating from COM to Add-Ins, Debugging, Localization and more.

## Authoring Add-Ins with .NET – Add-In Wizard



ESRI provides integrated development environment (IDE) templates for building and deploying these custom UI elements.

Users can quickly get an Add-In project running by selecting the components they want and supplying any necessary metadata including captions, categories, tooltips, descriptions, images, etc.

Once the solution is created, additional components may be added later using Add Project Item.

## Authoring Add-Ins – Best Practices

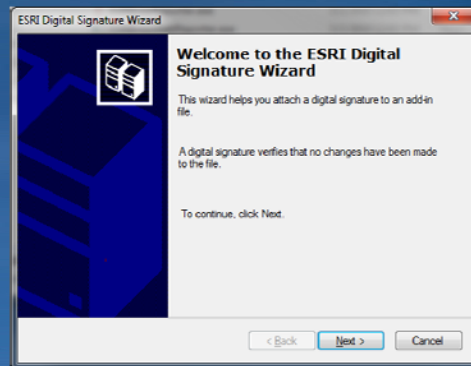
- Break the COM mindset
- Keep code out of buttons and forms
  - Centralize code within your extension
- Minimize work in OnUpdate
- Use “Load on Demand” pattern

## Authoring Add-Ins with .NET - Walkthrough



## Digitally Signing Add-Ins

- IETF/WC3 XML-DSig standard (within OPC archive)
- Trust
  - Source Traceability
  - Tampering
- ESRISignAddin Utility



The ESRISignAddIn utility supplied in the ArcGIS SDK can be used to sign ArcGIS Desktop Add-Ins. To use this utility, you must first have an ITU X.509 certificate containing both public and private encryption keys. Digital certificates can be issued by certificate authorities within an organization, or by a public certificate authority such as VeriSign or Thawte. Once the input Add-In file is selected, you will be prompted with a list of certificates you are authorized to sign with. Once a valid certificate is selected, the Add-In file is then signed and output; either overwriting the original, or under a new file name. The utility can also be used to view or remove existing digital signatures.

## Add-In Limitations

- More restrictive set of extensibility points
- Single user deployment model (unless using network share)
- Public API exposure
  - Can't register COM interfaces in registry
  - Can't register managed types in GAC
  - ✓ Dispatch mechanism for external access to simple properties and methods

List of supported types at beginning of deck.

Add-Ins are installed by simply placing them in a folder, and uninstalled by deleting the file from the folder. Add-Ins cannot rely on special setup steps or steps which would require administrator privileges on the machine (including any sort of COM registry, or registration within the GAC).

Exposing a public API generally requires actions which violate the previously mentioned restrictions. The Add-In framework defines a mechanism for very simple cases; all Add-In Extensions and Dockviews support this interface. Public properties and methods on the underlying class are automatically exposed through a language neutral Dispatch interface. This makes it possible to—for instance—call a .NET Add-In from a C++ client, or a Java Add-In from another .NET Add-In.

## Questions

## Further Reading

- Building Add-Ins for ArcGIS Desktop in SDK help
- WC3 XML Digital Signatures
  - <http://www.w3.org/Signature/>
- ISO/IEC 29500-2:2008 Open Packaging Conventions
  - [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=51459](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51459)
- *ITU X.509 Certificates*
  - <http://www.itu.int/rec/T-REC-X.509/en>