

# Migrating Custom Desktop Solutions from 9.3 to 10

Ralf Gottschalk and John Hauck show how to get your developer applications up and running within the enhanced framework of ArcGIS 10.

<http://video.esri.com/watch/654/migrating-custom-desktop-solutions-from-93-to-10>

---

## Video Transcription

**00:01** So, welcome to Migrating Custom Desktop Solutions from 9.3 to 10.

**00:04** This is kind of a whirlwind tour of migrating all of your customizations...

**00:10** ...including stand-alone applications, custom components, VBA, and VB 6.

**00:18** Are there any .NET developers who build custom components?

**00:24** A couple of you guys. How about stand-alone apps? Engine developers? Okay, cool.

**00:32** [Inaudible audience comment]

**00:34** To-be engine developer; good. VBA people? VB 6?

**00:41** Alright. Cool. Well, let's go ahead and get started.

**00:44** Let's save all the questions to the end, 'cause we've got a good amount of material to cover, I want to make sure we get through it all...

**00:50** ...and John and I will be available afterwards for...for any questions you have.

**00:56** My name is Ralf Gottschalk. I'm a product engineer on the Engine and the ArcGIS Runtime team. This is John Hauck.

**01:01** ...before you do any ArcObjects code.

**01:04** My name is John Hauck, and I work for the user advocacy group, as the tech lead for Desktop and Engine SDK.

**01:12** So first, we're thinking we're going to talk about is the changes at ArcGIS 10, your SDK requirements...

**01:18** ...and then we're just going to give a little note, 'cause a lot of people are interested in it, on...our 64-bit support.

**01:25** We're going to talk about publisher policy files, 'cause that's different at ArcGIS 10; there are no publisher policy files.

**01:32** We're going to talk a little bit about runtime binding; this is the step you have to do if you

build a stand-alone application...

**01:39** We're going to talk about what changed with component registration at 10.

**01:45** So you had a component that was sitting on your machine that just worked at every version; all of a sudden at 10...

**01:50** ...it didn't work anymore. Why? What changed? What do I need to do to get this working?

**01:56** We're going to give a brief introduction on the new customization framework options, add-ins, and the enhancements of Python...

**02:03** ...and those are things that VBA developers might want to look into, because they're good places to go...

**02:09** ...when you're trying to figure out how you're going to migrate your VBA solution over to a supported environment.

**02:16** And then we're going to briefly talk about VB 6 migration - what a VB 6 developer can do to start their migration process...

**02:23** ...to get everything over to a supported development environment.

**02:28** So, changes at ArcGIS 10. First off, the SDK requirements. You need to have Visual Studio 2008 SP1 or Visual Studio 2010.

**02:38** You can use Visual Studio Express 2008, not 2010, because 2010 requires, or is only for .NET 4.0.

**02:48** We've only certified ArcGIS 10 at .NET 3.5 SP1.

**02:53** At ArcGIS 10.1, we will certify for .NET 4.0, and also, just so you know, at ArcGIS 10.1, we'll only be supporting Visual Studio 2010...

**03:06** ...for the development environment. If you do Java, it's Eclipse or Ganymede, I mean Eclipse Galileo or Ganymede, and...

**03:16** ...we support at minimum JDK 6 update 16, and big change for developers...

**03:22** ...is that there's now a single ArcObjects SDK for each programming language.

**03:28** It's no longer an engine SDK, a desktop SDK, a server SDK - it's all just one, ArcObjects SDK.

**03:34** And if you build Engine applications, you have to, in your development machine...

**03:40** ...install the Engine Runtime first, then install the SDK, and then license the Engine Runtime.

**03:48** The change was made so that the only way you could build control-spaced applications...

**03:52** ...so Engine comes with these controls, they're visual components that show maps and whatever your custom forms are...

**03:59** ...in order to build applications with Engine, you need the Engine Runtime on your machine.

**04:05** But you don't need the Engine Runtime on the target machine you're deploying to. That works the same as it always has.

**04:14** Sixty-four bit support, obviously, Desktop and Engine are 32-bit applications, and they run as a 32-bit application on a 64-bit OS.

**04:23** So for stand-alone applications, if you're doing .NET development, you have to set your platform to X86.

**04:31** Individual Studio configuration manager, a lot of people don't do this, the default when you build a new form space application or...

**04:39** ...a console-based application is NECPU.

**04:41** What happens is, when you run an application, a .NET application on a 64-bit OS with the NECPU setting, it actually runs...

**04:48** ...as a 64-bit application. Our assemblies aren't 64 bit, so it tries to load the 64-bit versions of our assemblies...

**04:57** ...and it can't find them, and you get an error. So if you specify X86, your .NET executable will run as a 32-bit application.

**05:06** You don't need to do this for DLLs, 'cause DLLs take the...they inherit the process space that your parent application is running as.

**05:16** The thing we did for 64-bit OS is at ArcGIS 10, all of our applications are large address aware. So what this means is...

**05:24** ...a 32-bit application can only consume 2 gigs of RAM maximum.

**05:29** A 32-bit application running on a 64-bit OS can only consume 2 gigs of RAM maximum.

**05:35** A large address-aware application that's 32 bit sitting on a 64-bit OS can consume up to 4 gigs of RAM.

**05:43** So this is pretty helpful for doing your large geoprocessing tasks or just crunching lots of data.

**05:48** You can use more RAM, you won't get tiling as quickly, hopefully things finish for you, would be good.

**05:57** Publish or policy files. So, for .NET developers, we don't ship publish or policy files anymore.

**06:01** We want to add the Map Inquiry Identify command, and we see the GUID and the Prod ID here.

**06:03** What publish or policy files are is, when I create a .NET executable or DLL...

**06:10** ...my .NET executable or DLL has a specific version that it needs to run.

**06:16** So it...when it launches up, it looks in the GAC, the global assembly cache...

**06:21** ...finds those assemblies at that version. If it can't find those assemblies, it won't run.

**06:27** In the past, we ship publish or policy files. What that did was say, I built my app at 9.2.

**06:34** It tries to load on a 9.3 machine, it looks in the GAC, it can't see 9.3, but it sees a publish or policy file.

**06:40** And it goes, Oh, what does this say? This says, It's okay, just go ahead and use the 9.3 assemblies.

**06:45** So it would just redirect itself automatically and your component would just work.

**06:50** Is that a good idea? Probably not; you should probably recompile your component; we always recommend you recompile.

**06:56** In fact, you should really recompile your component for each version.

**07:01** At 10, we don't ship these policy files anymore.

**07:05** What we did do was we created, we added a configuration file to ArcMap executable, ArcCatalog executable.

**07:12** That configuration file redirects all 9.3x assemblies to 10, and it's there for your migration purposes, right?

**07:21** You can test your old DLLs, make sure that they work, and then recompile them and redeploy them.

**07:27** Runtime binding. So runtime binding is new at 10 because what runtime binding is, at ArcGIS 10...

**07:35** ...every product lives in its own install folder.

**07:39** Before, you'd install Desktop, and then if you installed Engine, they all shared the same bin folder.

**07:44** Now, they're separate. This is good, because now you can install your service packs separately.

**07:51** Before, if you wanted to upgrade to Service Pack 1 and you had a machine with Server and Desktop on it...

**07:55** ...you had to service pack both of them at the exact same time with the exact same service pack.

**08:00** Now they can be service packed separately, and you can go in and uninstall the service packs.

**08:04** So it's not a waste of time to try to see what Python can do for you. Because it's easy to use, it's easy to fix...

**08:05** Well, what that does to you now is that you have to tell ArcObjects where the bin folder is that you wanted to use.

**08:13** So all you do is this...the way you do this is just you bind to a runtime. And you have to do this for any stand-alone application.

**08:21** And you probably should do it in your main method, if you're a C# developer, or if you're a VB.NET developer...

**08:27** ...just do it in Application Events. John's going to demonstrate that in just a little bit.

**08:32** Do this before any calls to ArcObjects, otherwise you're going to see problems...

**08:37** ...and remember, binding is not licensing. Binding is just saying where ArcObjects are.

**08:42** Licensing is what level of functionality my ArcObjects can have.

**08:47** To do binding in .NET, you just add a reference to Esri.ArcGIS.version...

**08:53** ...then use the Runtime Manager static class called Bind, pass in the product code, so Engine Bind Desktop or do Engine or Desktop...

**09:04** ...and then there's also bind license method, which actually calls AO initialize under the hood...

**09:09** ...so it does your licensing and binding at the exact same time.

**09:13** So I'm going to pass it over to John, and he's going to do a quick demo of binding.

**09:21** Alright, so we've got a simple Engine application here.

**09:24** We have just a simple Windows form and a single map control and a license control.

**09:28** We have set up our licensing, but now we need to go ahead and take care of this binding. We'll see what we have to do to do that.

**09:36** We've added a reference to the esri.arcgis.version assembly, and now in our main method for this program class...

**09:45** ...we'll go ahead and make that call to the Runtime Manager, the bind method.

**09:51** Take a look at the code. There's one.

**10:06** Passing in the appropriate product code. In this case we also want to potentially bind to Engine or Desktop...

**10:12** ...depending upon our licensing. And that's all we really all we have to do to make this call to bind.

**10:20** So let's see what we would need to do for a VB application.

**10:30** Same type of simple Windows form, a single map control, and a single license control.

**10:35** But we know that we want to handle this at the appropriate location...

**10:38** ...so that way we can make sure that we bind before we attempt to use any ArcObjects.

**10:43** We can do that in the application events.

**10:46** We'll scroll down and pull up our application events. And then we'll stub out the start-up

method or event handler.

**10:56** And here we could handle our call to bind.

**11:03** So binding, you need to this for, again, just to reiterate, all stand-alone applications, even your Desktop ones, not just Engine's.

**11:12** So if you wrote a console application that does some geoprocessing somewhere...

**11:15** ...you're going to have to call bind before that application runs.

**11:19** Otherwise it can't find ArcObjects, and nothing's going to work, really.

**11:24** So what about custom components?

**11:28** Well, the only real change that happened at 10 with custom components is the old component registration step.

**11:35** So I'm going to try to explain how components were registered before, and those of you who already know...

**11:41** ...are pretty good COM developers, it might be very high level, but I think a lot of people don't really understand component registration...

**11:49** ...they got a DLL from somewhere, maybe they built a DLL to Visual Studio or VB 6 integration tools...

**11:55** ...which just kind of did this step for you; you've no idea what happened, really, and then all of a sudden it didn't work in 10 anymore.

**12:01** It's frustrating. What's going on? So let's go over what happens when you register a custom component really quick.

**12:09** So at, prior to 10, at 9x and 8.3, ArcGIS knew what kind of customizations it had on the machine...

**12:16** ...by looking in the registry in the component categories.

**12:21** So every custom component for ArcGIS has a component category.

**12:25** All my...all my commands are registered in the MxCommands category for ArcMap.

**12:29** All my...all my commands for ArcCatalog are registered in the GxCommands component category.

**12:38** So if I built a custom command, I would implement "I command," and then I would register my component after I built it.

**12:47** That component would also get registered into the MxCommands category.

**12:51** When ArcMap spins up, looks in the category, says, oh, those are all the buttons I need to load.

**12:57** It loads them all, and you've got your button in ArcMap.

**13:00** So just to kind of show this in a sort of a drawing, ArcMap loads up, looks in the registry, sees the MxCommands category.

**13:08** That category is pointing to my COM DLL that I've registered on my machine.

**13:12** That COM DLL then loads. It actually points to my DLL. That DLL then loads, and I've got my command in ArcMap.

**13:21** So when I run Register 32, if I'm doing C++ or VB 6 - well, VB 6 is different, but let's just say C++ for now...

**13:31** ...it would register both the category information and the component.

**13:35** When I run Regasm if I build .NET customizations, it would register the category information and the component.

**13:44** So what happened at 10? So at 10, we broke the products apart, right.

**13:49** We also had to remove the part of the ArcGIS that looks in the registry for the component categories.

**13:56** We don't look in there anymore. Instead now, we look in that ECFG file, this configuration file.

**14:03** That configuration file is the same thing as that category information.

**14:10** So now you use Esri Regasm to register your component.

**14:15** That Esri Regasm registers your custom component into the registry.

**14:21** Then, it creates this ECFG file. This ECFG file is sitting on your machine.

**14:26** When ArcMap spins up, it looks in the folder that contains all the ECFG files...

**14:32** ...sees what buttons it needs to load, and then loads your DLL and adds your component to ArcMap.

**14:39** So your components are still COM-based components, we just don't look in the registry for the category information anymore.

**14:46** So use Esri Regasm to register your components. This creates the ECFG file...

**14:51** ...and then it sticks it in a folder depending on what product you're registering your ECFG file for.

**14:57** So if you're registering it for Engine, it would stick it in the program files, common files, ArcGIS Engine 10 configuration CAT ID folder.

**15:06** If you're registering your component for Desktop, it's the same folder but for Desktop 10.0 configuration CAT ID.

**15:16** So to deploy your custom component, you have two options, really.

**15:20** You can call Regsvr32 or Regasm like you always did...

**15:27** ...if you used to - sorry. I jumped ahead. If you used to call Regsvr32 or Regasm on, in your setup when you deployed your component...

**15:36** ...or told the end user or whatever to run this step, you can still do that...

**15:42** ...I keep jumping to option 2. I'm sorry, guys.

**15:46** Option 1, if you used to do that, instead use Esri Regasm.

**15:50** And to use Esri Regasm, all you do is pass in the DLL, do a /p, put in the product you want to register it for...

**15:56** ...and the /s option is to do it silently.

**15:59** So this would actually register the DLL on your target machine and create the ECFG file.

**16:07** The other option you can do is register like you always have; use REGSERV 32 or your registry file.

**16:17** And then just copy the ECFG file from, on your developer machine, when you've created it...

**16:24** ...just copy it over to the client machine into the appropriate directory, and your component will just work then.

**16:32** You can also, if you're migrating an old project to 10, you also need to...the easiest way to debug that project is to...

**16:41** ...add this Esri Regasm step to your .NET solution, so to do this you just unload, edit the project...

**16:48** ...add the custom build step, and then this will register. We have this in the SDK, and John's going to show it too.

**16:55** This will register your component when you build it and unregister it when you clean your project...

**17:00** ...and if you're a Visual C++ developer, you can do it in a custom postbuild event...

**17:05** ...but there's no clean equivalent in the custom postbuild event.

**17:10** Real quick overview - John's going to go over all of this that I just talked about, and you'll see it in action...

**17:15** ...so it really will help with understanding, but to my great - your custom components - it's kind of like it's always been.

**17:22** We've included this ArcObjects in the ArcObjects SDK a code migration analyzer.

**17:27** It just runs through your code and tells you if there's any errors or things you need to do...

**17:33** ...to make sure that your code works in the next version.

**17:36** And then you just need to go in and update your debug start action in Visual Studio to change it, because remember...

**17:42** ...ArcGIS now lives in different bin folders at ArcGIS 10, so your old Visual Studio project's pointing to the old bin folder.

**17:50** You've got to just shift it over to the new bin folder...

**17:54** ...fix the changes - fix any changes to assemblies, add the new component registration step, and...

**17:59** ...if you're doing a stand-alone application, just bind, add the binding code.

**18:03** So let's see all this in action, and it might sound kind of complicated when I talk about it, but when John shows it...

**18:08** ...I think it - it'll be real easy.

**18:10** Alright, so I've got a 9.3.1 command or a command that I developed for 9.3.1, and I want to kind of step through the process...

**18:15** ...and so we can, you know, take a look at each of these individual steps and our options for creating this new ECFG file.

**18:23** So the first thing I want to do for my product here is run the code migration analyzer to see...

**18:27** ...if it lets me know about any issues that I may need to resolve.

**18:31** The first suggestion is that the solution platform should be set to X86.

**18:36** So this is just a - We don't necessarily have to do that for this command...

**18:41** ...but it's the best practice to do that, so let's do that first.

**18:45** We're going to go to Build, Configuration Manager. Currently we're targeting any CPU...

**18:51** ...we want to add a new one for X86.

**18:55** We'll just select it, X86, and this is just letting us know that it's going to copy from any CPU settings.

**19:03** So now we're targeting X86. The next one is to let us know that we need to replace this ADF assembly because...

**19:10** ...now we need to use ADF.local. So let's go ahead and handle that.

**19:17** Remove ADF, we'll add a reference to the ADF.local assembly.

**19:27** So we've taken care of all these...of the issues that we were informed of from the code migration analyzer.

**19:34** Now let's go ahead and try to build our project and see if there's anything else that we may need to do.

**19:40** Our build succeeded, but I want to go ahead and actually take a look at some of this code, and that way we have an idea what's really in here.

**19:48** Of a single rubber-band zoom class, this class is implementing a base tool, and a base tool is just a convenient way...

**19:57** ...for us to handle the implementation of the required interfaces to develop a tool without having to really know...

**20:03** ...what all of these interfaces are or what we may need to add.

**20:06** So base tool implementation is something that's just a convenient way to step in and do that in a .NET framework.

**20:12** Let's take a look at the COM registration functions.

**20:14** So here we're telling it that we still have this code that would have been stubbed out by using this base tool...

**20:23** ...but we still want it to be registered with COM, so we're going to leave all this code there...

**20:29** ...and then we just have a couple more things.

**20:31** We have the category that we want to see when we open up the Customize dialog in ArcMap.

**20:37** And this is still the same kind of code that would have been there for 9.3.1...

**20:40** ...and the caption that we want to see as we're browsing through these categories to try to find this command.

**20:46** So our build succeeded on this, we have...we have everything there that we had before...

**20:52** ...and I'm going to kind of pretend like I don't know about this custom registration step.

**20:56** So let's see what would happen if I don't use Esri Regasm; I continue to use the Regasm.

**21:04** So let's go to our folder where our DLL lives.

**21:11** We're going to go ahead and call Regasm.

**21:13** We're going to open up the Visual Studio command prompt, and I'll go ahead and register this DLL.

**21:24** My types are registered successfully; to me everything seems like it should be fine at this point.

**21:29** But if I were to open up ArcMap, I still wouldn't see that command listed in the Customize dialog.

**21:34** So one thing that we can do to get a better view into what is actually registered is we can take a look at the Component Category Manager.

**21:43** We can find that in our bin directory as well.

**21:45** So we'll go to C Program Files/ArcGIS Desktop 10/bin. And then we're going to look at the categories.exe.

**21:56** And in here we know that we wanted this to be registered with MxCommand, so let's take a look at that category...

**22:01** ...and see if we have anything registered.

**22:10** Scroll down to MxCommands, and there's nothing registered there.

**22:13** And we're not really sure why that happened, or we may have had at 9.3.1 a bunch of commands that were there...

**22:20** ...but they're no longer in this category. So a new button in this dialog at 10 is the Category Importer.

**22:25** This can be really helpful for finding assemblies that were previously registered...

**22:29** ...that, where we may not have had their solution files - we may have downloaded a DLL from ArcScripts, we don't know...

**22:35** ...where that DLL lives anymore. This can be a good way for us to go in and try to find some of these DLLs.

**22:41** The option that we have here is that we can select what we want to have registered and click Apply Changes.

**22:47** And when we do that, it's going to go ahead and run that Esri Regasm command for us.

**22:52** So this is a nice convenient option for people that have had components registered...

**22:57** ...that they can't find anymore.

**22:59** So let's see what happened in our CAD ID folder.

**23:07** So we're going to go to the Desktop install directory/configuration CAD ID.

**23:13** We see here that we have a new ECFG file that's been created for us.

**23:19** Another option that we have is that we could have run Esri Regasm manually.

**23:23** It would have done essentially the same thing - generate this configuration file.

**23:29** But I actually want to hand this solution off to another developer at our organization...

**23:36** ...and I don't want him to have to worry about the registration steps...

**23:40** ...and I don't want to package up the ECFG file and tell him where to copy it...

**23:44** ...so I'm going to go ahead and change my solution to automatically handle this for me on build and clean.

**23:51** So what we're going to do is, we're going to unload our project...

**23:59** ...we're going to edit the project, and down at the very bottom we're going to add these custom build commands.

**24:07** We have a knowledge base document for this that shows this code, so you don't have to try to remember all this or anything.

**24:12** We also have a blog that we put up with common resources for migration.

**24:26** So you see here, we're handling two different things.

**24:28** The first one is before clean, we want to go to the ArcGIS bin directory, grab our Esri Regasm executable...

**24:36** ...and we want to silently unregister with the product Desktop.

**24:42** Then we also have, after build, we'll go back to that same bin directory, so we can access this Esri Regasm executable...

**24:49** ...and we're going to silently register with the product Desktop.

**24:53** Now we've made our changes; we can go ahead and reload this project...

**24:59** ...now whenever we build, it'll automatically generate an ECFG file for us.

**25:09** If you take a look at the actual contents of these files, you'll see that...

**25:12** ...they're both, the configuration XML that's stored within here is exactly the same...

**25:16** ...it just has stores of GUID to the MxCommands category...

**25:20** ...and the GUID for our command that we want to have registered with that category.

**25:26** And if we clean, we would see that removed from the folder.

**25:35** Thanks, John. So John showed you the categories, the category executable.

**25:42** And that executable's pretty cool, because it allows you to see...

**25:46** ...what components were previously registered on your system.

**25:50** So both John and I - I used to work in support, John currently works in support, and I remember big problem when people...

**25:57** ...would migrate to the next version of the software with ArcScripts or other custom components...

**26:03** ...that just weren't really properly built that they had on their machine, they would install the new version of the software.

**26:09** The new version of the software doesn't know anything about your old components, so it doesn't remove them.

**26:15** They're just still there.

**26:16** And then you start up ArcMap, the old component loads, because it's all in the category information...

**26:22** ...and ArcMap blows up, because there's a problem.

**26:25** And it was always a pain to try to figure out, alright, which component broke ArcMap? Which ArcScript did this?

**26:32** With the categories executable, you can actually go look in the categories, and you can find the components that are registered...

**26:39** ...on your machine.

**26:40** And with the new registration, where we don't look in the Categories folder in the registry, instead we look at the ECFG file...

**26:46** ...we're basically giving you a clean slate, which is really cool.

**26:50** Then you can go in and manually add your old components for migration purposes only...

**26:55** ...'cause really you should have upgrade them, get them to the new version...

**26:58** ...but you can test them out, see if they work.

**27:00** If they work, you can go through the migration process, or...

**27:04** ...if you've got components that you're just so dependent on and you really need to migrate to 10.0...

**27:10** ...you can use them for now 'til you get the new version of the component.

**27:14** But we always recommend upgrade your component, because we make changes to the software, we fix bugs...

**27:21** ...so old components are not always going to be guaranteed to work in the next version, because they could be exploiting a bug...

**27:27** ...or maybe there's something else going on with them that just breaks, breaks you.

**27:33** So it's a good migration tool to try to get you moving forward.

**27:37** Real quick, on the new customization framework options, so these are focused in mostly on VBA developers, but...

**27:44** ...other developers might be interested in these, just because they make life a little easier in some respects.

**27:50** So a new customization framework option is add-ins. Add-ins do not require COM registration.

**27:56** All that stuff I talked about before doesn't apply to add-ins. Add-ins just load dynamically as your application loads.

**28:04** You don't have to register any components, you can use Java or .NET to build an add-in, and there's a wizard in Visual Studio...

**28:11** ...or if you do the Java version, there's a wizard in the Eclipse.

**28:14** And the wizard just, you click the wizard and it adds buttons...

**28:17** ...and it adds tools, and then you just type your code in for the click for your button.

**28:24** The wizard creates this Config.esriAddinsx XML file and a class for your button...

**28:31** ...and you can use the wizard to create new items, or you can go in and modify the XML.

**28:37** And the XML, once you start getting used to XML, it's actually really easy to work with.

**28:41** When you compile the add-in, it creates this Esri add-in file.

**28:46** So it's just a single file.

**28:47** It contains everything you need to run your add-in.

**28:50** And you just copy that onto somebody else's machine, and they can double-click it, and it installs your add-in.

**28:58** So the types of add-ins you can build are buttons and tools, combo boxes, context menus, toolbars, tool palettes...

**29:07** ...dockable windows, extensions, and editor extensions.

**29:09** So it's a restricted number of customization options you have in add-ins, and they're all related to the framework.

**29:16** You can use dockable windows if you want; you don't have to, you can still use forms...

**29:20** ...just load a form from a button.

**29:23** You can't create custom layers or custom feature classes or go down that road. We locked that out.

**29:31** Add-ins are just for framework-based customizations, but they're really easy to build, and really, really easy to deploy.

**29:38** They're installed on a per-user basis, so if you build custom components, it's kind of a pain, because you have to...

**29:44** ...make sure the guy has admin rights that's installing your custom component, because it's got to hit the registry and...

**29:49** ...add all the COM registration stuff.

**29:51** And then you have to deploy it, so you have to give them a DLL, you have to give them instructions.

**29:57** Add-ins are easy. You give them the add-in file. That's it.

**30:01** They don't have to be admins, because nothing gets installed in the registry...

**30:03** ...nothing hits your program files directory; it's all within the user.

**30:08** You can sign add-ins, and the system administrator can through the registry actually control add-ins, so...

**30:14** ...you can block them, block only the unsigned add-ins, specify a specific folder where add-ins can come from, or...

**30:24** ...only allow our add-ins on the machine.

**30:27** And you can also stick them on a network somewhere, which is really cool for VBA developers...

**30:32** ...'cause, I don't know if VBA people, if you've tried to deploy your VBA code, that's always been a pain too.

**30:38** Add-ins are great, you stick them on a network share, you go to the add-in configuration manager...

**30:43** ...point to that network share - the add-in will load automatically when ArcMap spins up.

**30:48** And if you make changes to it, you don't have to redeploy anything, you just update the network location.

**30:53** Next time ArcMap spins up, [it] knows about all the new changes, and your add-in is up-to-date.

**31:00** So I'm going to pass it over to John to show a demo on add-ins.

**31:07** Alright, so let's step through some of the processes that would be some of the normal things...

**31:11** ...that we're going to do when we're going to develop an add-in.

**31:13** And one of the things that we've seen is especially for VBA developers is switching the frameworks...

**31:19** ...go into a new development environment. There's always going to be a learning curve associated with that.

**31:23** But once they see kind of the fundamental elements that are involved, processes start to move really quickly.

**31:30** So let's take a look at some of this.

**31:32** We've started a new, we're going to start a new VB project of an ArcMap add-in.

**31:39** We can define some of the basic properties - the name of the add-in, company name, any of this basic information.

**31:48** But what we want to create here is a button. So we're going to give our button a class name...

**31:52** ...and this is going to be the name of the actual class in our VB file.

**31:55** Then we're going to give it a caption, and this is what we would see whenever we're browsing our Customize dialog in ArcMap...

**32:01** ...and once again the command, or the category that this will be registered under in that same Customize dialog.

**32:07** So we'll go ahead and click Finish.

**32:10** First thing we're going to see is our add-in's configuration file. And this is the basic structure of this add-in.

**32:20** So we - the majority of things we can add through our GUIs, but once we do that, we're not necessarily stuck...

**32:29** ...with whatever we added the first time around.

**32:30** We can easily go in here and change some of the information.

**32:33** We may want to change the caption, or maybe change company name, or give it a different version.

**32:38** So we can do all of that here in our configuration file.

**32:42** But what we want to do first is, we want to add an additional component.

**32:45** We want our command to open up a form, and we're going to start to build up some more content on that form.

**32:52** So the first thing we're going to do is, add a new form...

**32:59** ...on our form we want to go ahead and we're going to add a single button over here so we can kind of test out this framework...

**33:04** ...make sure everything's going okay - we can double-click this to stub out our event handler.

**33:10** And we're just going to show a message box.

**33:23** So we need a little information here saying something's still not right, we need to resolve this issue here, so...

**33:30** ...we'll go ahead and we'll allow Visual Studio to add our import statement.

**33:38** Now we need to handle the click event for our add-in. Let's view its code.

**33:45** And we just want to show that form. So this is something that's a little bit different than VBA, 'cause we need to...

**33:51** ...instantiate our form, so we're going to dimension a variable for form.

**33:58** We'll say it equals to a new form, and we just want to show that.

**34:13** So we also would like to go ahead and make use of some of the additional commands that are already available for us in ArcMap...

**34:22** ...some of the standard commands, zoom in, identify, and we want to add all of these to a toolbar.

[34:27](#) So let's see how we can do that here in our add-in project.

[34:31](#) We'll add another new item. We're going to use another ArcGIS template.

[34:39](#) But this time we want to create a command bar. So we're going to choose a toolbar, we're going to give it a name...

[34:46](#) ...and then we're also going to reference that button that we created earlier.

[34:54](#) In our configuration file, you'll see this added some new information.

[34:58](#) We have a toolbars collection because this add-in project could contain multiple toolbars.

[35:03](#) And then we have an items collection within that toolbar that because the toolbar could contain multiple buttons.

[35:09](#) So we'll go ahead - we actually want to reference another system command, and this time...

[35:13](#) ...we're going to do that directly here in our XML.

[35:18](#) We just need to add a reference ID to either the command's GUID or its Prod ID.

[35:26](#) We'll take a look at how we may be able to find that information.

[35:31](#) I've got the VBA help open here because I don't have a network connection, but, so...

[35:37](#) ...I'm going to add the Identify command. So let's search for that.

[35:59](#) So we found our command that we want to add.

[36:07](#) So we'll go ahead and grab this, and we'll add that to our XML.

[36:16](#) So really quickly here, we've been able to develop a custom toolbar, add a few commands to it...

[36:22](#) ...some that we'll handle code for, and some that we're just using from the system.

[36:33](#) Okay, so that's kind of a whirlwind tour of add-ins.

[36:37](#) But this is a good option for a lot of people, VBA-wise, to go to the framework components...

[36:42](#) ...or the exact same framework components as you had in VBA...

[36:46](#) ...and you can gradually start migrating your code over into add-ins...

[36:50](#) ...and really don't be scared of the XML, it's really easy once you start playing with it.

[36:54](#) It looks initially scary, but it isn't.

[36:58](#) Another option for developers at 10 is Python. So at 10, we've enhanced Python significantly.

[37:04](#) We've added this ArcPy site package. What this ArcPy site package allows you to do is...

[37:10](#) ...you get to print and export map documents directly from Python.

**37:16** So it gives you all this mapping automation functionality.

**37:20** Before, Python only gave you GIS analysis functionality. You could hit ArcToolbox, and you could do...

**37:28** ...you could run all the GP tools from Python.

**37:30** Now you can actually get access to your map documents and you could actually print.

**37:37** You could change layer symbology by importing a new layer file.

**37:38** You could fix broken data sources with Python.

**37:43** And you could automate the creation of map books, which is what a lot of people use VBA and VB 6 for.

**37:51** So given this, this might be a really good option for a lot of people in the future.

**37:55** In fact, even if you build custom components, and even if you have VB 6 code...

**38:00** ...maybe Python is a good move for you, because Python is a really easy API that you can use...

**38:07** ...that can be leveraged throughout the whole system. So all the time that you spend trying to build a Python script...

**38:14** ...you could actually use it in your code, in your add-in.

**38:18** You could use it in your custom component. It'll work on Desktop, Engine, Server, Windows, and Linux.

**38:31** ...you can interact with it in ArcMap - there's a Python window that you could bring up...

**38:36** ...you can type your scripts in real time, see what it's doing, and use it anywhere else.

**38:44** So I'm going to go ahead and just pass it over to John. [Unintelligible] He'll give us a demo of what you can do with...

**38:52** ...Python and ArcGIS 10.

**38:57** Okay. So we're going to take a quick look at how we may start to interact with Python at 10.

**39:05** So the first thing that we want to do is we want to go ahead and export our map to a PDF.

**39:13** So let's take a look at what may be involved with that. To do that we'll go ahead and look at the Desktop help.

**39:19** In our professional library under Geoprocessing, we can get access to the contents about the ArcPy site package.

**39:27** Specifically, we're going to be looking at the mapping module and some of the functions that it provides.

**39:31** We want to export to a PDF.

**39:34** We have a nice discussion of what this method provides us, along with the syntax that we would use to invoke this command.

**39:43** So we're - the notation here is that if it's with the curly braces, then it's optional.

**39:48** So you still have a lot of optional parameters, to really do some fine-grained control over this export operation.

**39:55** But the only required methods are a map document and a path to the output PDF.

**40:01** So the first thing that we need to do is get access to a map document.

**40:04** When we're working in ArcMap, we have access to the current map, so we're going to call ArcPy...

**40:13** ...we're going to go to the mapping, and we're going to open up or get access to our map document.

**40:22** We're going to use this current keyword.

**40:28** So we have access to our map, and now let's go ahead and call our export function.

**40:32** So we'll do ArcPy, mapping, and we're going to export to PDF, passing in our map and the path to the PDF.

**40:55** I don't know if any of you wrote export code in ArcObjects, it's definitely more involved than a two-line export operation...

**41:04** ...But - so a lot of times whenever you're working with Python, you're going to be doing some quick testing...

**41:08** ...you want to maybe interact with the current map, get an idea of what you may be able to do...

**41:13** ...and then we can go ahead and save from this window to a Python script.

**41:25** Now after we've done that, we want to go ahead and open it up in Python and make some additional changes to it...

**41:30** ...so we're going open up PythonWin, and we're going to load in that script.

**41:41** One thing to note is that when you are working in the Python window in ArcMap...

**41:45** ...it handles the import of ArcPy for us, but we need to handle that in our code.

**41:52** But we want to actually expose this as a standard geoprocessing tool...

**41:55** ...so that way we can take advantage of it anywhere where we could take advantage of standard geoprocessing tools...

**42:00** ...in ArcMap, in our .NET add-in, or wherever.

**42:04** So we want to allow the user to provide us a path to our MXD.

**42:19** Then, also since we're not working in a current map document necessarily, we're going to have to provide this path, so...

**42:26** ...we'll go ahead and put that here.

**42:30** And then we just want our PDF to write to the same directory as our MXD...

**42:35** ...so we can use this same map path that we got from the user.

**42:41** And we're just going to pull off MXD and replace it with PDF.

**42:53** So now I have my script all set up.

**42:56** I went ahead and loaded this into a toolbox, so that way we could open up ArcToolbox and use it there...

**43:03** ...but we actually want to make use of this in our add-in. So let's see what's involved with that.

**43:12** Got my add-in project open; we're going to add an additional button.

**43:21** Go back to this familiar set of dialogs and define all of these standard properties.

**43:29** And then we can go ahead and start coding out our interaction with this geoprocessing tool.

**43:46** Add some [unintelligible].

**44:09** So the first thing that we need to do is start to take care of some of this stuff, and in this case we have no...

**44:17** ...there's no - it doesn't start to suggest...

**44:24** ...I probably am missing a reference in the project to this assembly.

**44:28** So I could take this, and then I could go out to the Resource Center, look up that assembly...

**44:33** ...and at the very top of the Resource page it will list out the assembly that this comes from.

**44:38** I happen to know that it comes from the ArcMap UI assembly, so I can go ahead and add that.

**44:44** ...that it can automatically add my import statement for me, so that lets me know that...

**44:52** Oh, I'm sorry. That's the carto assembly. So I'll add a reference to carto, and then it will automatically handle this import for me.

**45:05** I have another one here where it's still not qualifying, and I've already provided the full path to the class...

**45:12** ...so I know this one for sure is missing its assembly reference, so I'm going to go ahead and take care of that as well.

**45:32** As soon as I add my assembly reference, I don't have to go ahead and tell it to add that import statement...

**45:36** ...because I am using the fully qualified name.

**45:40** And the last one we already have a reference to, so we can once again just allow it to add our import statement for us.

**45:47** So what we have going on here is, we're hooking into the application using the static class called ArcMap...

**45:53** ...and getting access to our document.

**45:56** We instantiate our geoprocessor and add a reference to the location of the toolbox that contains our Script tool.

**46:04** We add the parameters that we need, in this case we're going to be pulling this from a map document...

**46:08** ...and we're just going to pass in the path to this document file.

**46:12** Then whenever I added the script to the toolbox, I defined its name as Export to PDF...

**46:16** ...so whenever I call Execute, I'll just pass in the name of my tool and the parameters that are associated with this tool.

**46:23** Then I could easily take advantage of some of the functions that are provided through ArcPy.

**46:29** What ... the big side of that, I guess, is that we really want to - we don't want people to avoid going down the Python road, because...

**46:37** ...it is somewhat limited in scope right now because there are a lot of convenience things that are added to that...

**46:44** ...migration patterns, especially, for VBA developers.

**46:47** It provides a lot of unique pathways.

**46:49** One of the unique aspects of it, whenever 10.1 comes out and we have add-ins for Python, this provides an opportunity to...

**47:00** ...Python is not a compiled language, so whenever I make changes to the code that's associated with my UI command...

**47:07** ...I don't have to recompile that, so I can leave ArcMap up and running while I go ahead and make changes.

**47:12** This kind of thing doesn't happen for when we develop DLLs.

**47:16** If I'm developing an add-in, and I want to make some additional changes to it...

**47:20** ...I need to stop ArcMap, make the changes, recompile my DLL, restart ArcMap.

**47:24** And so there's a little bit of overhead there, and some - this is something unique about - that Python will provide.

**47:33** Thanks, John. So you can see Python is pretty powerful, you can do a lot of stuff, and...

**47:37** ...it's probably a really good solution to check out, at least, and see if maybe Python is what

you need to do your work.

[47:47](#) We're going to talk real quick here about VBA migration strategies.

[47:51](#) So, first off, at 10.1, what's the status of VBA?

[47:55](#) At 10.1, we're going to provide a free VBA compatibility setup.

[48:00](#) Just like at 10, you have to request a license from us. It's a free license, you don't have to pay for it, but...

[48:05](#) ...you need to get the license and then you run the setup. VBA's on your system.

[48:11](#) We're not going to provide technical support for VBA, and we're not going to give you an SDK.

[48:16](#) So we kind of threatened you before, saying 10.0 was the last release for VBA.

[48:21](#) It's not, there's VBA still in 10.1.

[48:24](#) This is good, 'cause it gives you a lot more time to migrate over from VBA.

[48:29](#) We're not taking VBA away because we're mean and we don't like you and we don't like VBA.

[48:35](#) I love VBA. I learned ArcObjects on VBA. But we just can't support it any more.

[48:40](#) VBA's going away at some point. We don't know when. We don't know if the next Windows is going to support VBA.

[48:47](#) We can't really realistically support it, so we're giving you, kind of, here's VBA, here's VBA, and then one day...

[48:55](#) ...VBA won't be there anymore, and it's not really going to be our fault, why it's not there.

[48:58](#) So it's just - you have to start thinking about migrating away from VBA.

[49:05](#) But at 10.1 your VBA stuff will still work.

[49:08](#) So we really want you to migrate. Where does the VBA guy go? I mean, really, that's a hard question to answer.

[49:13](#) And I don't have a good answer for you, other than, you have to look at your workflow, you have to look at...

[49:18](#) ...what you're doing with VBA and then you can decide where you want to go with VBA.

[49:24](#) So Python's an option. If you're doing lots of map automation, or geoprocessing, with VBA, then Python...

[49:33](#) ... is the way to go. It's a simple API. I mean, Python even gives you access to cursors and stuff like that...

[49:37](#) ...so you can actually work with features directly, you can work with geometries.

[49:42](#) It's really easy to program in Python. Once you start playing with it, it's way easier than

ArcObjects...

**49:48** ...it's way easier than VBA.

**49:51** And over time, we'll have more and more functionality with Python. So it's going to get bigger and bigger...

**49:55** ...and, like John showed you, you can use Python in any development environment.

**50:00** So your investment into Python is not lost if you decide that Python isn't good enough for you.

**50:07** 'So why is it not an option, potentially? Well, obviously, it's not ArcObjects.

**50:11** You can't do what you can do with ArcObjects with Python.

**50:15** So you want to look through the ArcPy doc, see what you can do, see if it fits your needs...

**50:20** ...maybe part of your solution can leverage Python, and the other part might be an add-in or something like that.

**50:25** And the other problem is, yeah, you're going to have to completely rewrite your code.

**50:28** You can't just - It's Python. It's a different language, completely. But that's why it's easy to use.

**50:36** The other potential option for VBA developers is add-ins. So why would you migrate to an add-in?

**50:42** Well, you can copy and paste a lot of code from VB right into VB.NET, and Visual Studio just kind of like solves stuff for you.

**50:52** So it's not too hard to go from VBA to add-ins.

**50:57** And the other thing about add-ins is, if you build custom components and you have to deal with all this registration stuff...

**51:02** ...it's confusing, going from VBA to component registration, it's a leap, I mean, it's a challenge.

**51:08** Once you understand it, it's no big deal. But add-ins provide that nice intermediary step...

**51:14** ...where you don't have to worry about that. Just get my code in there, and let's get stuff working again.

**51:18** It's the same type of customizations, and it's so easy to deploy. That's really the big thing.

**51:24** And when you are migrating your VBA code, what you want to do is, if, let's say you have a bunch of VBA forms, or...

**51:31** ...just a bunch of document-specific VBA things, you want to start abstracting that.

**51:36** Pull it out, stick it in a DLL, or stick it in an add-in. And then start using that, and call that DLL from your form...

**51:44** ...or call that DLL from your VBA customization, and then start migrating forms over.

**51:52** That would probably be the best way, because you need to pull all your code out.

**51:55** I mean, the one problem with add-ins, it's not VBA, so it's not as easy as VBA, and there is no per-document solution with that.

**52:06** VBA, you would just open up ArcMap, be like, for this map document, I want to do this, this, and this, and this.

**52:11** Add-ins, you have to make them more generic. It's not document-specific.

**52:15** So let's go ahead and pass it over to John, and...

**52:17** ...he's going to show a quick demo on taking some VBA code and putting it in an add-in.

**52:23** Alright, so I have a command here that when we launch it, it's going to go through a single field in one of our layers...

**52:30** ...and list out all the values, and we can use this to select a feature on the map.

**52:38** Nice and simple. But let's take a look at the steps, common steps, that are going to be involved with this type of process...

**52:44** ...of bringing this form and everything over into the add-in world.

**52:49** So the first thing we need to do is take a look at our VBA code.

**52:55** I'm going to kind of move this screen around a little bit here so you can see.

**53:02** So I've developed a similar form to what I had in my VBA, and this is actually the form that we added...

**53:10** ...in the initial add-in demo, but I've added a combo box, and now we need to, on our form load...

**53:18** ...we need to add this code right here.

**53:22** So in .NET, we can double-click on this, automatically stub that code out for us.

**53:28** We'll paste our code in, we'll start to take a look at some of the stuff that's really involved here, but...

**53:32** ...things that you notice immediately are, it automatically removed all of those set commands.

**53:36** So some of this redundant stuff, I don't necessarily have to do just because I'm moving code from one language to another...

**53:42** ...and there are minor syntax differences.

**53:45** Also, there's this interesting notation with VBA where if you surround something with parentheses...

**53:52** ...you have to surround something with parentheses if it's going to return a value, but you don't if it's not...

[53:57](#) ...and this is another type of issue that's automatically resolved by copy and paste.

[54:03](#) Then I want to move over the code from the Click command on that form.

[54:14](#) Then I have a single subprocedure here that we'll also need to move over.

[54:24](#) Now we can go ahead and try to start taking care of the issues that we have here.

[54:29](#) The first thing that we see here is that we're missing another reference, so we'll, I mean import, so we'll go ahead and add that.

[54:41](#) We also need to take care of this issue in VBA, we had access to this document...

[54:45](#) ...it was a nice, easy hook into the document, but with add-ins we have a static class called ArcMap, so we'll use that again.

[54:55](#) And this just provides us a nice, easy hook into the document.

[55:02](#) Next we need to, we'll go ahead and just continue on this path of just adding these import statements.

[55:09](#) This one, once again, I would take this interface, take it out to the Resource Center, punch it in, and...

[55:15](#) ...I could get on my API reference doc, it'll tell me the assembly that this comes from...

[55:19](#) ...but I know it comes from geodatabase, so I'll go ahead and add that.

[55:32](#) And then now, even though we've added our reference, we still need to add this import...

[55:37](#) ...and even though we'll have a lot of common, the control types between what was available for us to add to our forms...

[55:47](#) ...in VBA are going to be similar to the control types that we have access to for our .NET forms...

[55:53](#) ...some of how we start to interact with these may be a little bit different...

[55:57](#) ...and that's the case here; we have a combo box, but now we need to add our items in a little bit different way.

[56:02](#) So we go `ComboBox.Items.Add`.

[56:09](#) Now we're going to add all of those values in this loop to our combo box.

[56:14](#) Once again, we need to hook our document, so we'll use that ArcMap class again.

[56:24](#) Then the next thing that we see is, for a suggestion, it lets us know that we need to add a reference to the ArcGIS geometry assembly.

[56:35](#) You see that this one is a little bit different in that it's allowing me an option to automatically add this, and this happens because...

[56:42](#) ...the method that we're calling exists in an assembly that we already have a reference to.

**56:47** That assembly knows additional assemblies that it's dependent upon, so in that case...

**56:52** ...it can automatically give us a little bit more of an easy way to automatically add that.

**56:57** So we'll add our geometry reference, and then another little trick that you may be able to use is that, a lot of times...

**57:04** ...if you just go to the last element in your statement, it can start to kind of give you some options...

**57:12** ... on how you may go about resolving that. In this case, just passing in the constant from the enumeration isn't enough.

**57:19** We need to go ahead and use the full enumeration name. So we'll go ahead and add that.

**57:33** We'll need to do the same thing down here. Once again, we can go back to that last element and get a suggestion.

**57:53** Same thing.

**58:01** This stuff's obviously a simplified demo, but these are a lot of the common things that you're going to have to do...

**58:07** ...whenever you're moving your VBA code over, and if you know where to look in the developing environment...

**58:12** ...if Visual Studio's not familiar to you already, and you kind of get some ideas of where it's trying to help you...

**58:19** ...understand what needs to happen, then it will hopefully help provide some insight on how you can start to do...

**58:26** ...some of the changes that are going to be necessary.

**58:30** Cool. Thanks, John. So that was VBA, taking some VBA code, sticking it into Visual Studio...

**58:38** ...and you could see, Visual Studio does a lot to really kind of help you migrate.

**58:42** Yeah, it's going to take some time, it might be a little frustrating, but the documentation is out there...

**58:48** ...you can always go on Google and you can always find information that will help you migrate stuff over. We're going to...

**58:55** ...I didn't see a lot of VB 6 developers earlier when I raised my hand, but we're going to really quickly go over what's going on...

**59:01** ...with VB 6 at 10.1. So at 10, actually, as of 10, VB 6 is no longer a supported platform.

**59:09** We won't ship an SDK, and we don't install the VB 6 runtime.

**59:13** This is important, because newer versions of Windows don't install the full VB 6 runtime, so if you have old...

**59:19** ...VB 6 components that require certain aspects of the VB 6 runtime, they're not going to work on new Windows systems...

**59:25** ...and we're not going to install it for you. You're going to have to install that yourself.

**59:31** So really we want you to migrate away from VB 6. Migrate to a supported language, and we're not going to go into...

**59:38** ...the details of how the migration goes, because the ArcObjects SDK has tons of information in there on migrating.

**59:45** It gives case studies, samples, all these tips, it's really good on migrating VB 6 code, and there's no way...

**59:52** ...we could do all the things we just talked about and actually show you how to migrate VB 6 in addition.

**1:00:00** But I want to tell you why your VB 6 components don't work anymore. Why didn't they work?

**1:00:05** So people got kind of mad at us, because they had these VB 6 components that were on their machine...

**1:00:11** ...they maybe got it from ArcScripts, they hired a developer or something, or maybe they built it themselves...

**1:00:15** ...they've had it on their machine for who knows how long.

**1:00:18** And then they start up 10 and VB 6 doesn't work anymore. Why? Why is VB 6 not working?

**1:00:23** And then try to use Esri Regasm on VB 6. It doesn't work. Why is it not working?

**1:00:28** Well, it's kind of part of the nature of VB 6 itself. So, earlier, when you saw John building a custom tool...

**1:00:35** ...there's this COM registration portion in the code in Visual Studio.

**1:00:40** What happens in a .NET and a C++ DLL is this COM registration stuff actually executes when you register your DLL.

**1:00:49** So I do register 32 or I do Regasm. Code runs. That's why you have to have admin privileges to register a DLL...

**1:00:57** ...because code runs. You could stick some...

**1:00:59** ...You could be mean and you could stick something nasty in there, and if you really wanted to - hopefully nobody wants to...

**1:01:06** ...But when that code runs, it actually registers that component on the machine in the categories...

**1:01:11** ...and it registers the component in the registry, like we talked about earlier.

**1:01:15** VB 6 can't do that. So it doesn't register anything in the categories. So when ArcMap spins up and it looks in the categories...

**1:01:22** ...whatever your VB 6 stuff is, it's not there. So most people use reg files to register VB 6 stuff.

**1:01:29** Or they did add from file in ArcMap, and that got the information in there.

**1:01:34** So that's why Esri Regasm doesn't work with VB 6 right out of the box, as it stands. But it can work with VB 6...

**1:01:42** ...and I'm only telling you this because I know there's a lot of people who probably have...

**1:01:47** ...tons of VB 6 DLLs and tons of VB 6 components that they really want to use, and...

**1:01:52** ...they don't have the time to migrate all of them right now.

**1:01:55** They want to migrate all of them , and they will migrate all of them...

**1:01:58** ...but it's going to take time, because people really relied heavily on VB 6.

**1:02:03** With VB 6, it's the same story as with VBA, I mean, it's not like we just dropped it because we hate VB 6.

**1:02:08** We dropped it because we have to, I mean, it's time to...

**1:02:12** ...VB 6, you can't even buy the VB 6 development environment anymore.

**1:02:17** So we don't block VB 6 components from working in ArcGIS...

**1:02:21** ...we just...VB 6...we don't look in the categories anymore to load the component.

**1:02:25** So you can actually register a VB 6 component, for migration purposes only, to test your components...

**1:02:33** ...and to slowly move forward to a new development environment.

**1:02:37** We have a KB doc out there that tells you what you need to do to migrate, to actually register a VB 6 component.

**1:02:43** We can't guarantee it's going to work, because...

**1:02:45** ...you're supposed to recompile every component at every new version of our software.

**1:02:50** It probably will work, because it's COM, and usually it just works, but you really need to test it...

**1:02:56** ...and don't use this as your business solution. But use it as a migration; gradually replace your components.

**1:03:04** That way you can still use them, and that way you can move forward and get your work done.

**1:03:09** So I'm going to pass it over to John - he's going to show you this real quick.

**1:03:14** Okay, so I have a reg file here that I've used to register this command in the past.

**1:03:20** I'm going to take a look at what's going on in there.

**1:03:22** This is pretty similar, actually, to the config.XML that's stored inside the ECFG file.

**1:03:31** We keep track of the GUID of the category that we want to register it with.

**1:03:35** And then the GUID of the command that we want to register with that category.

**1:03:41** Then I have a simple install script.

**1:03:43** And all I'm doing here is similar to what we've seen in other examples so far...

**1:03:47** ...is going to our bin directory, grabbing the Esri Regasm executable.

**1:03:51** We're going to register with the product Desktop...

**1:03:53** ...but now we get into a little bit different arguments for actually handling the VB 6 registration.

**1:04:02** So we're going to use the MyCustomTool DLL that's sitting in this folder, and then...

**1:04:07** ...we're going to pass in the path to our reg file. So this will be uploaded online as a kind of a template, and...

**1:04:14** ...I believe there's some stuff in the - I think the KB doc goes over that as well, so...

**1:04:19** ...there's a lot of information on how you can do this same type of workflow.

**1:04:32** Yeah, just - oh - good thing we're done with the presentation, because apparently we're having some...projector...

**1:04:46** I was just going to load up the KB doc, but just to show you - it gives the - there it is. It gives detailed instructions on what you need to do...

**1:05:00** ...to get this component registered, and it even shows you, if you're not familiar with how to create a registry file, it kind of has a sample.

**1:05:08** And that's it.