PRESENTER 1: So I'm going to turn this over to Tiny, and he's going to talk about deciding what is a time to automate and not automate.

PRESENTER 2: Thank you, Jerry. Everybody can hear me OK? All right. So through many years of doing a lot of automation scripting and stuff like that, this is my company's perspective on when we get automation requests. General rule of thumb is we tend to look for commercial. And if there's something, a commercial product out there, or a vetted third party application out there, that does at least some of what we want-- 60%, 70%, 90%, whatever you can live with-- we always tend to go with that commercial product.

And the reason behind that is, a lot of people may say, OK, well, commercial products are very expensive. And we've got some in-house programmers or developers, and stuff like that. But you really have to look at the whole cost associated with automating. You have labor associated with the costs, you have purchasing and licensing. One thing that a lot of people don't look at is your production staff. When you're talking about automating, if it's outside of the physical IT or CAD environment, and you're looking at workflow automation or stuff like that, then you actually have to take highly paid engineers to come into a conference room and discuss the process and go through all that.

And then a lot of people don't think about the residual effect of the automation as far as training and the support, bringing your users of the speed, and then also the compatibility as far as for the changing environment. So really think about, when you go into automation-- like I said, try to look for a commercial product or a valid third party.

Some keywords that we're going to kind of go over as the process-- when you think, OK, now, I've already checked out, there's no commercial product out there that I want to do. It's very specific to my environment, or whatever. A few things to make your automation successful, or a few things to look at and then also go back and say, well, we might not even need to automate. We probably should forget about this.

As one, before you even start defining your goals-- and I've been on many scenarios where I'm getting feedback during the automation process, and the goals just change left and right. And you go down one path, and then another goal comes up, and you have to look and re-look at how you coded it, or what kind of tools you're using, and say, well, that's not going to fit

in this scenario. So now we've got to go backtrack and figure it out again. So definitely get some real defined goals.

So your automation is coherent later down the road. You might have to get another person to pick up from where your original person took off. And if that coding and that flow of the automation is not very coherent, it's going to be very difficult for them to pick it up. Plan. Plan before you code. Plan before you start developing. Plan. I mean, this is getting together in workrooms, discussing the goals, figuring out your resources, figuring out who needs to be part of it. Plan it. Because like I said, you can just keep going down the rabbit hole forever if you don't do some very solid planning.

And the planning, again, can cause you to redevelop, re-code, scrap a lot of stuff, and come back.

Time. Calculate what it's going to take to get to a final product. And then calculate how much time you're actually saving. And I know I've had coding requests or automation requests for one thing. And it was like, I could spend six months doing this, and what is it going to do? Save an hour or a week, or something like that? Is it worth it? And the six months is just to get the initial product out there. And then from there on, you have the maintenance and everything going forward.

And then your return on investment. Definitely calculate the labor. Think about the labor. Think about the purchase. Think about the maintenance. And when we're talking about automation, And this is the next bullet. It's simple. When you automate, keep it simple. And I don't know, how many are software development company in here?

All right, so a lot of us might be more of the engineering side or the industry side, and stuff like that. And we just want to get these simple routines automated so we can get rid of the mundane tasks that we have to do. So just remember, we're not really paying for-purchase software. So when you start looking at your goals and your plans and stuff, if it's starting to get so complex, revert back to that original thought-- let's find a vendor. Find a vendor to do this.

Make sure you're flexible with the technique you use. And if you're going to automate a process that you do on a daily basis, keep it flexible, because down the road the goals may change. And if the goals change or if you want to reuse that later, you want to make it where it's very flexible to plug into other scenarios.

Reusability. So basically, the longevity of the automation that you're actually developing is what's going to bring you the most return on investment. So if you actually develop some form of automation or automated routine, and it's only valid for a six month period, you really have to look at what was that time savings, what was that ROI?

But if I can develop a simple routine that can save somebody one to 10 hours a week for years to come-- and we're focusing a lot of this on a lot of the licensing and deployment. And that strategy hasn't changed very much in many, many, many years. So if you develop an automated technique, that could be a very long-lived benefit.

And like I said, for me, I like to develop automation for those mundane tasks. If you're doing a manual mundane task, that's when you need it. I would suggest look at that. Don't look at trying, like, ooh, it would be cool to have this tool in an all new environment, and stuff like that. That's software development. Automating is what you already do. You just want to make automated.

Sources. Make sure whatever sources you use are well vetted. They're credible sources. It's not like a downloaded software that might tie into some process, workflow process that you have, or some extraction tool that might be used for data extraction-- MySQL or something like that.

If you start building your automation on that, and then that source no longer supports the code, no longer supports the tool, Windows environment changes, then your automation is broke. That goes back to that long-lived value of actually automating in the first place. And Jerry and I both agree a lot of times. I say, freeware is nice for home environments and stuff like that. But freeware can become expensive-ware, in the sense if you start using one, it's no longer supported, no longer functional, you've got to go back to the drawing board again.

So on this slide here, we wanted to bring to your attention that a lot of the stuff that we're going to go over today, you already have. You don't have to go out and buy something then do an automated technique or automate a process that we're going to present on. But don't forget about the simplest tools that you have already available. From licensing lmutils and the command prompt world, the pinging, the IP configs, the registry queries and stuff like that. Depends on what you want to automate, but you may already have something on your desktop ready to go and then you just massage that and get it to actually work for you.

And in all the automation, all the VB programs I'm going to show you and stuff like that, I'll use

that same technique. Never bought a piece of software, never had to buy a third party add on to do the things that I actually do.

And Jerry, you want to pipe in on that?

PRESENTER 1: Yeah. Like the slide says here, the bottom line, I think, is very important. Don't forget batch files. You're going to see a lot of really fancy VB stuff that Tiny's done, nice GUI interfaces. And that's OK. But for a lot of stuff, you just need fast and dirty. The batch file's really valuable, because anything that's up there can be automated with a batch file and be done quickly, in hours, not days. OK. And then the only thing I'll say there is, once you did that, test the heck out of it before you really start using it.

PRESENTER 2: Yeah, so like I said, hopefully all those key words and stuff like that, for whatever you're going to automate on-- I know the title of the class is for CAD managers. So we're going to give you a lot of automated functions within the licensing and deployment side of things. But use the same practices for any automation.

So we're going to go over the Active Directory integration. Just for everybody's benefit, on the class sign-up, you have the documents and stuff like that. We've posted two of these tools out there for you to use. So this tool that we're going to go through for you to use-- it's freeware. And it's hopefully not going to be expensive-ware for you. It's been functioning for a good probably eight years, 10 years, now.

So basically what this tool is going to do is, it's a VB program, 2010 coding. It reads Active Directory. And it reads your Autodesk License Options file, which everybody should have. Correct, Jerry? If you've got a licensed server out there for Autodesk, you should have an options file.

And what it's going to do is read the options file. And with an option file, you can control borrowing and restrictions with it. And using groups-- and you use host groups, and you can use user groups. So what the tool is going to do is go out the Active Directory, read your options file, compare it to-- find like for like-- the groups, and pull those members out, or those hosts out, and then populate your options files for you. And this saves me a ton of time. And I don't have to worry about RDP into 10 or 12 license servers. I have this running all the time. If I need to add a member to a permission group, or to a borrow less, just add them in AD. They're done. All my servers are already synchronized and all.

So this is a good little tool. I hope that you find a benefit for it. The set up is pretty simple, too. You either create the groups in AD or you use existing groups that you already have. And then you're going to update the INI file. That basically tells it where these files are located, what domain you're actually on.

As I kind of go to re-usability, my automation is not written to my domain. It's shareable, goes to any domain. And I'll create a schedule task, actually, on the server. So I'll actually process everything for you.

So basically, steps to implementation-- identify and create your AD groups. Create a mirror group in the opt file, the options file. There should be one in on every one of your license files. So I actually have an opt file that is across the board. And so I push that out to all my license servers. And they have these groups and the permissions associated with it.

And then you run, you edit up the INI file, which we're going to show you how to do it. And then you're going to run the exe that actually updates the options file. Then you're going to check to make sure it basically populated the members. And then if it's good, you're going to create a schedule task and then you're going to test it again.

So from the download site, you're going to download update opt. And it's going to have an executable, it's going to have an INI, and it's going to have a PDF. You're going to open up this INI file. And it's really just basically for flexibility, where you install your lmutils, where you install your Autodesk manager. You actually tell it. And I'm giving it a reread, because after it updates the options file, your license server needs to reread it so it actually applies all your changes.

So I actually put in a port number and the computer name, which is universal. So it doesn't matter what PC or what license server you put it on. It'll actually run. Now the port number, you'll go in there and change if you're not using that port and stuff like that. And then also tell it where the options file is. Where is it located? And in the A to Z class where we teach it, we'll point out what we suggest as far as locations. But if that's not the case, you just edit this INI file.

And then the domains. What domain are you on? And then-- and this is why I actually made it flexible, not so I could share it with everybody, but because we go through a ton of acquisitions. And we ended up where we have four different domains. So the AD groups, I can have the one company that we just purchased or something like that go and create their own

AD group, and then synchronize it, so they can control their users for borrow. We can control our users for borrow. And it actually allows for a multi-domain environment.

In just the AD, looking at AD users in computers, I have a borrow group here in the Americas, BORROW-T1. It's a legacy borrow group. And I'm sure some of you might know what it means, but it's a legacy. One thing I want to point out is the tool understands that a group in AD can have another group, and that group can have another group. So in this case, I have a AM-BORROW-T3. And in there, I add the T1 group and the T2 group, so I don't have to keep maintaining that third tier type group. So the tool will read the group as it's reading the members. It'll find if there is another group out there, and it'll read that members list. And if it finds another group, it'll keep on going and populate for you.

So if we look at the actual Autodesk options file, which everybody should have, mainly because of these two lines here, time out and rolling of the report logs, keeping up with your report logs. I have a group and the name of the group, and I just put "noone." And this is when I stand up another license server, I just throw this options file out there. And then I set up my schedule task, I run it, and the "noone" should be changed and get populated with all the members associated with it.

Now the reason you want to create groups is so that you can actually control the borrowing. So this T1 group can borrow AutoCAD 2015 applications licenses. You can use it for borrowing, you can use it for restrictions and all. So how you use the group doesn't matter. It's just the name of a group that is associated with Active Directory.

So once you've actually edited the INI-- you got your groups, you got your members and stuff like that-- you double-click on the exe. And it actually runs, and then hopefully you open up the options file. And I'll show you what you'll see is, you'll see a bunch of names. OK. If that worked, as it should, next is actually starting up your schedule task. And when you go through a schedule task, it's very basic. Choose that you're going to run it on a daily basis. You're going to start a program. You're going to come in here and point it to the exe. The one thing you want to make sure is to give it the start.ini folder. So basically, you copy this first part and drop it in there. And that's mainly so the exe knows where the INI file is. And so wherever the exe runs, the INI file should actually be.

And then before you actually finish creating a schedule task, tell it to open up for advanced properties, because you're going to make a couple of other edits here. Here you're going to

tell it a user to log on and use. And this should be a service count. If you're license server administrators, you really should have a service account running that you can use to run these schedule tasks and all. And you just tell it, basically, to run this schedule task whether the user is logged on or not.

And then after that, under your triggers, you're going to edit the trigger, which you chose was the daily. You have the option of how often you want to repeat it. So this is just totally up to you, how often you change your borrow users. If you got a lot of influx on people saying, hey, I need to borrow a license or something like that, and you do it on a daily basis, then make it daily or whatever. So it all depends on how often you do it or you make changes to the borrow list.

Now after you do that, test it, because the difference between double clicking an exe and running a schedule task can actually be quite different. Because the account that you use, you might have mistyped the password. The account you use might not have access to the folder where your lmutil is located and stuff like that. So you want to test it. After testing, whether you run the exe or the scheduled task, this is what you should see. You're going to see multiple rows, depending on how many users you got in the borrow groups. So if you notice on the T1, it's only got one line. So we've got a few people that have a very high level of borrowing or restrictions, or non-restrictions.

And then T2, we actually have about three rows. And then T3, we got a ton of rows. Don't worry about that, because what the tool is doing is, once you exceed a certain limit-- the license manager tends to flake out and not like that. So the actual executable is maxing the characters per line by 2000. And the license manager knows, as I go through here, it's not the typical "the last one wins." It's actually concatenating a list over and over and over.

So that's what it actually should do. The executable that we're providing you also only does changes. And I didn't have this in earlier version, and my logs grew crazy. Because every time it would run, it would do an lmreread, and it would make the logs super huge. So I modified the code to go in and compare what the file was before and what it is now. And only if there was a change will it actually do an lmreread. OK.

So that's actually the tool that I give you. Now you're saying, OK, well, kind of like that freeware is expensive-ware. Or maybe we don't want to use Tiny's tool and stuff, which is perfectly fine. I just want to show you there are other techniques that you could integrate with Active

Directory or pull data out of Active Directory and not use this actual tool. There are some complex ones and you really need a pretty savvy way of understanding AD some scripting and some syntax. And then there's a lot of really easy ways.

So here are three examples that just come with the OS that you can actually run. It will pull data from Active Directory. One is table oriented, one is comma separated, CSV format. And then there's just a query service. And so basically the goal is a tool to extract AD group members into a text file that you can then go put in your options file. So that's the ultimate goal.

So the one I found that was the simplest to actually use, where you didn't need to know a whole lot of information other than your group name, is dsquery. So you're doing a dsquery group, the sam account ID name, and this is all you need to know. So if you take this line, get members, and then process the sam account IDs, you take that line, put it in a bat file, and then pipe that to a text file, you're going to get a list of users, usernames, which you actually need, or the host names which you need, to add to your options file.

So now once you get it, it's going to give you a table output. So then you just need a secondary process to go in, because the option file requires the user names or the host names to be separated by a space in a linear fashion. And that can be done pretty much with any other type of scripting and stuff like that. So I'm going to turn it over to Jerry. He's got to talk about one tool that we both love. But Jerry is the master of it, and he loves it the most. It's lmutil.

PRESENTER 1: I guess it's because I started life in the industry as a Unix administrator in the days before there was much GUI there. So I spent my entire life at a command line. It feels warm and fuzzy to me.

So everybody that has a license manager gets a copy of lmutil. It's just a command line executable. There's a lot of power in it, so we're only going to scratch the surface of it in this class. There's a document that comes with the license manager. It installs in the folder where you install the license manager. It's a PDF file. I strongly recommend that you go in there some day and actually do the boring thing and read chapter 12. It could really inspire you. Hey, I was trying to do this over and over and over again. It was really hard. I'm clicking on lmtools, and I only see the little window. So read through this thing. You might find other possibilities than we're talking about that are of use lmutil.

And what I do typically on my workstation-- and I don't really usually execute lmutil on the server. I execute lmutil on my workstation. And I create a folder on my computer. I call mine bin. Once again, it makes me feel good because it sounds like Unix. But you can call it anything you want. But I put all the different utilities that I use, all the executables that I use, in there, and add that to my command path. So then I don't have to switch into a particular folder to run those things. Just makes life a little easier.

I really hate typing. I like command line, because I can see what's going on, but I don't really love typing. So this cuts down on how much typing I've got to do.

So I'm going to go over a few examples. lmnewlog is special because you can't do this in lmtools. There is no function like lmnewlog in lmtools at all. And what this does is it archives the report log, the .rl. Who records RLs on their license server? Good chunk of you, OK. So if you have flexnet manager, flexnet manager will automatically rotate that log file for you so you don't have to touch it. But if you're using SAMReport-lite, or if you're just simply building the report log for a second source of truth against something else, which a lot of people do, then you've got this file that's growing. And like anything else that grows, as it gets too big, it starts causing problems.

So the idea is to take this script that I've written, run it on the server once a month using a schedule task like Tiny just showed you for his Active Directory integration. So we do the same thing with this. And we run this lmnewlog once a month, is typical, and it builds a monthly archive. And if you look at the example here that I've got, it's lmutil lmnewlog function. Minus c says pointing to where the license is. OK? And I'm saying, you'll find it on @127.0.0.1. Now if you talk to any Flexera person, the guys who work for Flexera, makes ths stuff, they get mad when they see this. Oh, you shouldn't, you shouldn't do this. But this particular is a special situation. You can calm them down, because what this is designed to do is, I can take the script just the way it is and run it on any machine.

The only time it doesn't work is if I happen to be, say, in the UK. They use a different time and date format. So when I parse the date out here, it scrambles it. So you've got to go in there and adjust the parsing there. That's all. So when you wrote the first time, if it looks like gibberish, the file name looks like gibberish, it's because you're on a non-US date and time format. And once you stare at it for a few minutes, it's pretty easy to adjust the parsing for what you have.

Now, there are some accounts that you'll run into where if I save this is a batch file and try to schedule the Task Manager, it's going to generate a bunch of errors saying your service account doesn't have run as batch permissions. And if that occurs, and your IT security people don't want to allow you to give the service that permission, what you can do is instead you schedule lmutil. And then and that Task Scheduler, it says, what parameters do you want me to pass to that program? And you just give the rest of the line as a parameter. And then you've gotten around the fact it doesn't want to run a batch file.

I use this one as an example, because it's also got a gotcha. So the typical use case for this is people write scripts that use lmreread, particularly when they have, say, a centralized bank of option files. And you'll do the edit the options file on a central machine. And then they'd write a script that copies the options file out to their servers and then does an automated reread. So it's one edit, one button to reread, and they're done. What's the gotcha? Anybody know?

The gotcha is, if one of these rereads fails, and it's an automated script, and the command window opens and closes, you don't see the error. So you assume all three of these servers got reread and have the new information you got. And one them failed. That's why if I'm doing this interactively, I just go ahead and I put a pause at the end of my batch file. And I can look to see what happened. If I'm not doing that, I want to do one of several things. I can always go in and track for errors, error codes. After each one of these, do an if and see if I have an error condition, do something else, like send an email or whatever so you know what happened. Or you got to go and look at the debug or server status to make sure that reread really occurred.

Because it's not uncommon for a network glitch to kill a reread. It'll work again in two minutes from now, but it's not uncommon for reread to fail.

Now this is the most powerful part of lmutil. And this is what I use day in and day out in my work and Tiny uses in his. And a lot of Tiny's automation is based on information that we pull out of the license manager using lmstat. And essentially, it provides you information on licenses and the health of the license server. It has some flags you can use-- you have to use, so the most typical one is minus a. It basically dumps all the information out. It's very verbose.

Minus s is specific to a given vendor, say, adskflex, which is a Autodesk vendor daemon. I seldom use the minus s because usually I fixe the port number that lmgrd runs on for every vendor in a license server. So if I've got brand x and brand y and all this licensing, I put them

all in specific ports so I can find them reliably. So what I'll do is I'll say, minus a, well, I'll do a minus a. And when I specify where the license is, I say 27,002 at whatever the server name is. OK? And then this way I don't have to bother with the s. Same thing. Minus f gives you feature specific information.

I personally don't generally deal with this either. I just dump everything a, dump it to a text file, and then massage the data that I need to see it once it's in the text file. That's generally how I do it. Tiny does it otherwise. Furthermore, the minus i gives you a nice neat looking table. It's a fixed column table. Tiny likes to use it for a lot of things, and he's got a couple of good reasons to do it. I'm going to let him talk to this particular part of the lmutil in a minute.

I don't use it. What I use instead to get my columns of information about what's in the license file, I use the license parser. Anybody use licenseparser.com Check it out if you haven't. It's an Autodesk site, actually. And it still says to this day, it's been out there for what, about five years Tiny? And it still says it's beta, but it's there.

So the lmstat minus a gives you everything. And I'm going to kind of quick quickly go through this. Same thing for this one. And F what I want to get to is here just to talk a little bit about the output that you see. So basically, I didn't know this is an lmutil, lmstat minus a and pipe that to a text file. So I can search on any of this information in here. So if I can parse the information out the way I want to see it.

A couple of things that are important. You've got the vendor daemon version right, here you've got the master daemon version up here. So if you're required to have a certain version, it's a quick check for you. And here you see what really loaded, not necessarily a file that on your server. I've seen many times somebody accidentally configure the license server at a different group of files than what they thought they were doing, because they had two instances of the software installed. One was installed, when it was copied. And they just configured the wrong one. Here you see what's really happening.

The feature, of course, this is the license that you're actually trying to use. And probably one of the most important things is, who's got the license borrowed? Once again if you're running flexnet manager, that will tell you. But if you're not using flexnet manager, which is Flexera's high end reporting mechanism, if you're not using that, you don't know who's got anything borrowed. You don't know if anybody's got something borrowed unless you're looking for this linker flag. And I actually produce a report that's just a quick script file. I run this thing. I filter on

the word linger, and I get a list of everybody who's borrowed something.

And then a lot of people don't realize this. This crazy looking number at the end is the number of seconds that the license is borrowed for. So simple arithmetic, you can use to calculate borrow time in something that's usable like days. And then over here, you've got the user, the PC, the license server, the port, time and date that it got started. All available information.

Ideally, unless it's a borrowed license, you don't want to see anything that's not today in this column. Can anybody tell me how you avoid having what happen? Something older than today? Licensing use?

AUDIENCE: [INAUDIBLE]

Well, no. You can just set the time out all in your options file. And time out all will pull them all back in if someone's not using it. So in a moment I'm going to turn this over to Tiny to tell you how he uses the i switch on the lmstat. Like I said, I don't use it. I virtually never use it. Tiny uses it a lot, to let him talk to this particular slide.

Thank you, Jerry. So basically, the dot i is-- and I'll show you an output that it actually generates-- one, easy to parse. One, if you're doing scripting and automation, the option to have a licenseparser.com give you feedback and stuff like that is kind of difficult.

But licenseparser.com is a great tool. I use it the same way everybody else is. Used to take my 10 or 12 licenses and drop it on there and export it and pull out the features and populate my options file. It just got real hectic doing that, for me.

So I started using the -i because I could parse this out and get all my features, and then I could easily populate my options file that way. Also, it will tell you your license count. It also will tell you if there's an expired license out there-- a very easy, quick identity there. Go there.

So this leads up to we've talked about lmutils, switches, and stuff like that. Here's another tool that I've posted out there that you can download, the freeware version. And basically, it's a flexLM utility.

So if you are a license manager, an Autodesk license manager here, you probably have some time where OK, I need to know what my license servers are doing, I need to know to check the statuses and stuff like that. And you might have multiple license servers. In my case, I don't want to have to go RDP into every single one of them and run a report.

And also, what I wrote was, again, a VB GUI, and all it is is a shell. Everything it's doing is just what we taught you. We're running the lmutil with those parameters, and it's feeding back information that's usable, and it's feeding it to multiple servers all at once. So it's going to concatenate the reports and bringing back information for you.

Now, what I have is on the GUI, you have your FLEXlm license servers. You can go in here and specify the port at, and then license server name. If you don't use ports in your license file, which you should not do, or it's strongly suggested to use the ports, you can just put @ and then your license server name. And then you got a plus or minus so you can add that to it.

Next is actually the feature list, and if you-- if anybody knows real quickly anything different about the feature list here? One is in order by the product, because I stripped off the first five numbers. And all. So the GUI tool, originally I just had the whole feature name there. Well, every time I wanted to run a report on, I needed to know how many AutoCADs I got out there. And I use-- I had to go here, scroll down, go here, scroll down.

So the tool automatically pulls all the unique numbers on the feature code, so that you get all the products in the list in alphabetical order. In the tool, there's multiple-- this is just a drop down for the report you want to run. And up here is the reports. I have it categorized as custom reports and FLEXlm reports. The only difference is all of these are running the same FLEXlm, lmutil parameters. The only difference is I'm taking the output and parsing it inti a fixed width text file that you can view and easily read.

And also-- nothing magical going on here. The one thing that you'll see, there's a little trim here is, you'll have a total license by server. And then you have total license by server selected. And then you have license by feature, licenses by feature selected.

Any time you choose the selected one, it will open up this, where you can actually select not only which servers you want to run it against-- multiple or just one-- also which features you want to run it against. So if you want to see, I want to know the total licenses on this new server. I just got a new license file for all these features, or I just bumped up my AutoCAD. I want to know what the new license count is, it's easy to go through and run it and get a snapshot of that.

Some of the most useful ones I found is licensing use by feature, licensing use by user, or borrow. How many people has got the licenses borrowed? Who's got it specifically? So it runs

a lmutil again, the dash A command. Reads it, populate it, finds the linger, just like [INAUDIBLE] was talking about, parses it out and puts it in a readable format.

One little nifty one I actually like, and I made specifically. Because every time we get license files, I have to go in there and drop them in a location that populates all my servers and stuff like that. But I used to run to licenseparser.com to populate this tool with all the new features.

So this tool is run by, again, an INI file. And at the bottom it's got a list of features, and it's got all of them with that five character number. And I used to wipe it out and paste it in there. Wipe it out and paste it in there. And I was like, well, I can just use the dash-i. Run it after I've already updated all my license servers. And it gives me all the unique-- I can just parse out all the features.

And it's going to automatically update this GUI for you. And it automatically pull out just the unique features that you have in all of your license files. And then if you want to take that and export it out and send it to another spreadsheet you might have, doing some automation calculations or usage data or anything like that, you can easily export that out into a list.

Again, this is the Verbose raw report, the flex report. So you can do the same thing, but what you're going to do is notepad-- every time you run this report, notepad is going pop open and show you the report. This is going to pop open notepad, and it's going to have that very verbose language that FLEXlm utility is going to give you.

You can also run information on your vendor demons and your lmutil version. So you know you just went through an upgrade. You went to 2017. You updated all your license servers. I got 10 or 12 or 20 license servers. I don't know how many you got. I need to know, did I do them all right? Did I miss one? Did somebody not do their task? I gave it out to all the regions and stuff like that. I can just highlight them all, run that utility, and it'll tell me what versions on all the servers there are.

One thing I did as specifically for this class is extracted data. Because what I do is I run the lmutil command. It dumps out that very Verbose information. I process it and pull out whatever I want out of it. And I created in a CSV format a nonsummarize CSV. And then I go through and summarize it and pull out unique values and total values and stuff like that.

So if you just want the raw CSV without all that Verbose information, you can click here, and it'll pop open notepad with a CSV with all the data out of your license files.

This is just an example of one of the reports. In the reading guide for the class sign-up sheet and stuff like that, I had a number of these snapshots. Just for sake of time, I didn't include them all in here. So this is the licensing use by feature, listing all the current users.

So here's the servers. And I ran it on maybe all 10 servers. But these were the only ones maybe that had licenses. So you can see, and that can give you a clue, hey, I ran it on five servers. I really should see five servers listed here. Are one of my servers down? And why is there no usage on my server?

Also here is a asterisk to indicate that this is a borrowed event. And I calculate those seconds. This is when that borrowed event should expire. Because that's how long they borrowed it for. And also, if you haven't controlled your borrow limit, if you see somebody's borrowed it for six months, you might want to talk to that individual.

And so it gives you all the information. And here is this user is actually coming off of this server. If you go down here, this user is actually coming off of this server. So you know where they're pulling their data, and if you're across time zones, and you don't want that, across regions, that's a good indicator. Why are these users, or why are these PCs in Asia, hitting the server when they should be hitting a different server? So that's free. It's out there for you.

Now I'm going to turn it over to Jerry about the desktop setups.

PRESENTER 2: Thank you, Tiny. Well, this thing's got a sticky key every now and then. So it's often desirable to figure out what's on your desktop, as we spoke before. I'm going to talk about some of the key pieces of data you can query off of this thing and collect.

Environment variables, of course, are important. And also stored in a similar area on the registry is a thing we call breadcrumbs. Who knows what a breadcrumb is in FLEX talk? Breadcrumb is basically an area in a registry that FLEX stores the last, not just the last, but a list of successful license pulls.

So it says, I pulled a license from server a, I'm going to add them to this list. Pull license from server b, adds it to the list. It sounds really cool. It gives the software a strong will to live. Because even if someone clobbers the environment variable or something, you're still able to start the product. And everything's really hunky dory until you don't want Joe get to that server anymore.

You clobber his environment variable, and lo and behold, you still see him in there when you look at the log file. It's because there's a breadcrumb. So while it's a good thing sometimes, it's good for rote, it's good for medium sized and small environments. It really is good. It's very handy.

And for an enterprise, it can be a real headache. And just learning how to deal with them and knowing that they're there is important. So I'm going to talk a little bit about those.

We also want to be able to make sure when we're querying the desktops to figure out what we want to do with them, we want to make sure they've got good solid communication back to the license server, right? So we're going to talk a little bit about a couple of different methods to do that.

And then you quite often want to know the insulation type, whether standalone or network. And maybe the serial number that got installed, particularly for standalone, for compliance reasons. You don't want serial numbers on your network that you don't own, right?

So the ADSKFLEX_LICENSE_FILE variable is the variable used to point to multiple servers. And of course, it follows the order that they're in. After, by the way, it looks at the breadcrumb. It looks at the breadcrumb first.

And the FLEXLM_TIMEOUT is the other most common desktop environment variable. And that's used if you have latent connections to the license server from workstations. Careful with this environment variable, because you set it too big and it impacts-- well, it always impacts startup performance. So every time you make that value bigger to fix a problem as far as latency goes, it also takes longer to start the product.

So if you set this thing to 7 million, which is seven seconds, it might fix your startup problem. But you know, if you've got to go through five license servers, that's 5 times 7, that feels like eternity. So just use the smallest value that's possible if you have to use it.

The other piece about environment variables, what you want to be on the lookout for, is you don't want anything as a user environment variable, you want everybody using the user environment variable. And if you detect user environment variables, you want to get rid of them. They cause real problems with down road automation because, if you have both a system and a user environment variable, the user value wins, and you're not going to be able to direct your users to the resources that you want to direct them to. So you want to have a

very low tolerance of people setting a user environment variable.

Tiny put this in here. Breadcrumbs can be a pain. Once again, he has a huge environment, and he's got to deal with those things. And they are a pain for him. We'll talk about what to do about them in a second.

So I want to show you where they are and where they aren't. Because we're going to tell you to delete them, and we want to make sure you don't delete the wrong thing. So if you see the word software, it's a breadcrumb probably. So if it's HKCU, HKLM/Software, and you find a value for ADSK_FLEX_LICENSE_FILE, it's probably a breadcrumb.

If you find the same value under System, current control set, what is that? Anybody tell me? It's an environment variable. So if it was HKLM, it's the system. If it's HKCU, it's the user environment variable. If you catch one of those, you want to have a talk with that user, and tell them, please don't do that. Because it breaks automation.

If you find the breadcrumb registry keys, you can safely delete them. It'll get rebuilt. The next time you start the product, it's going to grab that next successful server and stick it in there. But you can get rid of the stale ones you don't want people to see.

Now if you've got a good solid active directory environment, you can use group policies to kill that breadcrumb when they log in all the time. And a lot of my clients do that in big environments. They just kill that bread crumb every time they start a work session.

Well a word about ports. You've got to make sure they're open. And you've got to be kind of aware of where things are going. Particularly if you're fixing a port, and you fix it between 27,000 through 27,009 in your license file, if you don't do anything at the desktop except point to the server name, it's going to find it. Because it's going to scan all those ports. If you go outside of this range, there's no scanning.

So if you set your manager license file to port number 29009, and you don't put that in your environment variable, you're not going to find the licensor. But the thing that we normally see is that the port that we're running on, either the 2080 or one of these, is being blocked somewhere in your network or by a firewall.

So there's a different method you can use. Of course, pinging is only going to see if the server is online. You can tell telnet the server port. What I like to use is the lmutil command. I'll just put a copy of lmutil on that workstation I'm trying to test, and do an LMstat-c in the server

name, and maybe the port number if I'm specifying that already in the license file. Because this is going to use the same basic communication that the software uses to go out and contact the server and ask for a license. So it's very real world. Tiny's going to show you another method of checking ports as well.

So I think we've kind of talked a little bit about this. Except that one important piece you got to realize is this whole thing with our deployment wizard. So if you use the deployment wizard, which most of you, I'm sure you do, to build your deployments to push out to the workstations. If you select distributed server, it's going to create an ADSKFLEX_LICENSE_FILE environment variable on your workstation upon install.

But it's got a gotcha. What's the gotcha?

AUDIENCE: [INAUDIBLE]

PRESENTER 2: The gotcha is that you can't specify the port to look at. So if you went through the trouble of fixing the port of your license manager 27,004, for example. And now you go start this thing up on a workstation, and you don't have enough port set at the workstation, it's going to start scanning at 27,000 until it finally gets to 27,004. Wasted bits and cycles.

So a way to fix that is, when you're building your deployments, lie. Tell it you only got one server. OK? This way the deployment wizard is not going to mess with your environment variable. It's going to create a like path at LIC in the executable folder of the application with that one server you put in there.

You can think of it as a spare tire. If the environment variable gets blown away, it will go back to that server. And the environment variable override and precedents the like path at LIC so by setting that. So outside of the deployment environment, you push out an environment variable with the port you want the people to go to, or ports, depending on servers. I find that's a much better way to handle a situation in anything from medium to large environments where you have lots of servers and you want to control the port access.

As far as doing that, as far as delivering it to the desktop, once again, if AD is stable and complete in your environment, which I know, a lot of people are challenged with. When you have lots of acquisitions and stuff, sometimes it's all scrambled in an organization. But if it's solid, this is by far the easiest and most reliable because you can get rebuilt every time the guy logs in. OK?

There's other methods though. The next favorite one for me is using setx.exe. Who's familiar with sets.exe? No? OK. It showed up in Windows 7, and it's been around since Windows 7 on the desktop operating systems.

It's kind of like the set command, except if you use a set command to set an environment variable, it's just for that session, that Dos window even. OK? When you do setx. and you set an environment variable, it's live for the entire computer right away. You don't have to log in, log out. Nothing. The only thing it's not is if you had a DOS prompt open already, of course, that won't see it. But any new future DOS prompts or anything like that are going to see that very variable. So it's a quick and dirty way of setting something manual.

And what I did was I wrote a DOS prompt, a DOS batch file that does this. And the one thing you got to be aware of is, if you do that, if you make a batch file, if you don't right click and say, Run As Administrator, it fails on anything Windows 7 and later.

So what I do is I trap for an error, because you right click and do that. And you don't see it if you just run a batch file. So I trap for an error, and if there's an error condition, then I say, try it again, and right click on it. Right click and Run as Administrator.

You can use other kinds of batch files. You can use VB scripts. I've seen people do-- another very popular method, if people have a more solid SCCM implementation than they do Active Directory, what they'll do is they'll push a reg key for the environment variable out via a SCCM. You know, that's great, except the user has to to log in and out for that thing to be live. And that's a pain. And that's why I like setx. so much.

But I have not been successful in getting setx. to run as a SCCM push. So the reg key seems to be the way to do it if you're running SCCM to push it. And once again, if you're using GPO in particular, it's so easy to clobber the breadcrumbs. Consider doing it, if you're--

So I'm going to let Tiny talk a little bit to this slide. This is basically the guts of a script that he wrote that queries the desktop. So he pushes it out for SCCM and sends the information back to a central location. So he basically can get an inventory of his network and what's going on out there. Tiny, I'll let you take it from here.

PRESENTER 1: Thanks. Yeah, so basically, kind of rolling back to what we just discussed as far as how to set the variables. You want to check the variables. So this is just a very simple script, batch file

that you can throw in. Use any push technology that you want to use.

But more or less all it's going to do is output the values of these variables to a text file and we're creating a text file and then we're appending. And we're appending and appending. So the first is just getting the variables, the environment variables, the FLEXLM.

We're using a reg query to actually search the software Autodesk folder. And we're looking for a standalone network type variable or value. And it's going to actually search the whole subfolder. So what it's going to do is going to dump out a list of all your applications that actually have this key. And that key actually tells you whether you have a standalone or network version.

It changes from year to year and something like that. So we also-- there's a couple of them that actually can run. License type. Standalone network type. So you can actually add multiple lines to this. But the same thing with the serial number. If you're concerned about people installing software that is not using the right serial number, if you have standalone licensing and stuff like that, do a reg query.

The port query is a command line tool that you actually can download off a Microsoft site. You're basically saying test this server. It's going to use TCP protocol on 2080, which is the Autodesk vendor daemon. And it's just basically going to check to see if it's open. You're going to get a report like this.

So it will say, OK, here's my server. Port 2080 is listening. And that's what you want to look for. And like 27,000 is listening. Now they got you on here. And that's why definitely you want to go back into an lmutil, and you can put that as your last line, lmutil-s or -a and port it to the same file. Because this might be another Brand X product listening. And it might not be the vendor daemon for Autodesk.

So if you have servers that actually have multiple vendor daemons installed on it, that one could have taken over that port, the 27,000 specifically. So you want to make sure that you're talking to the Autodesk vendor daemon. So use your lmutil also.

There we go. So I'm going to kind of breeze through this. I couldn't give you this, because I couldn't strip out a lot of stuff that's proprietary to our environment. But it's just another way-- it's using, this is actually using a remote registry technique that VB offers. And basically I can put a PC name in here, and I can choose and say, show me the license server availability on

that PC. That actually runs a remote port test. And it can tell me whether it passed or failed and off of the servers.

This one over here. I can also choose, check the PC license environment variable setups for this PC. It's going to go through, and here's all these keys. Now this is a lot of legacy keys that I found in my environment. And it basically will tell me that Autodesk license file, the LM project that we use in LM time out. So it will tell me whether it passed or failed.

Now some of these passed. They shouldn't-- from our presentation you should know, you really should clean these up. But I think one year, a product, even though you had it in the system environmental variable, sometime that product didn't really want to honor it. So it was kind of like, well, let's just populate it in every location that we find and all, so--

And then lastly is checking the PC Autodesk install. So I can put in a PC name. Dump it out. If you have SCCM, you're going to have some of this is real time, going out there querying the information. But it mainly-- if somebody calls me, says, hey, my AutoCAD's not working, I can check it. I say, well, your IT guy installed it wrong. He just ran setup that exe. And he didn't run it from the deployment and all, and it's standalone.

Another little port tester that I wrote as far as just a GUI interface, I had a lot of people, I'd just tell them, hey, run this port tester. Put in this value. Put in the port, and do check now. And it actually just gives them more of a visual. It's open. Again, it doesn't say that Autodesk is listening. It's just that there's some service running on the other side and it's listening on that port.

OK, so we're going to talk a little bit about the uninstall Autodesk applications.

PRESENTER 2: Thank you, Tiny. So when you create your deployments, you're going to notice a folder underneath the main deployment folder called SMS/SCCM. I don't know if you recall seeing it or not. But that particular folder contains some information that's actually useful even beyond SMS and SCCM.

So we want to share with you, not just the fact that you can use SCCM to install or uninstall programs, but also we'll give you a couple suggestions on other things you can do with that. Today we're going to concentrate on the uninstall part, because last year in Beyond Z, we talked more about how to get the software out to the desktop. And this year we wanted to talk about something a little different.

Basically the methodology is the same, but we're going to concentrate on uninstall today. The particular files that we're going to deal with is, the image name, whatever it is, if you called it ACAD_R2016. At it'll be that, underscore Uninstall.txt.

This is the actual batch file that you'd be running. But it's still named .txt at this point. And then, the ReadMe is basically telling you how to use this file. Because it doesn't come ready to run. If you take this thing and you run it the way it is, just rename it to bat. It's going to do nothing. It's actually a fairly common support call. People say, hey, I found it. I renamed it. It doesn't do anything. Well, it was designed that way.

So just to show you where it is. This is the 64ACAD17 was the image name. And underneath the image, you got the SCCM SMS SCM scripts, and your files over here. OK? The two ReadMe's. And this one pushes the software out, and this one takes the software off the workstations.

So when you open up that uninstall script, everything has been commented out. All the different features are commented out. So that's why it doesn't uninstall anything. The reason why it's been done is we want to force you to pick specifically what you want to uninstall.

Quite often if you're running multiple Autodesk products, you don't want to uninstall everything that the deployment put in there, because some of that became shared with other applications that you might have installed afterwards. So you just don't want to go clobber everything. You want to be specific.

So for that reason, you've got to go in and remove the double sets of colons in front of any of the items that you want to uninstall. And this is a really important one. You're taking software off a computer, OK? And if you take something off that's being shared, you're going to break other stuff that's sitting on a computer that you want it to continue to work.

So hopefully you've got a staging area, and you can test the batch file on some staging areas, or at least some users that are a, not doing something mission critical, and b, have a little bit of patience for something to go wrong, before you roll it out on a production basis. Because this one can have nasty effects if it hits the wrong person on the wrong day.

And here I'm going to let Tiny talk about-- On that same subject, Tiny kind of mocked up the sample file and showed you what he decided to include and not include in an uninstall routine. Tiny, it's all yours.

PRESENTER 1: Thank you. Yeah, so basically, and I don't know how many of you knew about the uninstall.txt file and stuff like that. The first year they introduced it, it did not include what is included now. So I made that same mistake the first year. I opened it up, I find, replace, colon, colon, and replaced it with nothing. Uninstall. Worked perfectly fine for 2015.

2016, 2017 didn't work that way because the structure of the file-- kind of move over here and show you, this is the Civil 3D sample text file. So in yellow, now the bat file or the .txt file Includes the platform applications. So you're installing Civil 3D, the platform for Civil 3D is AutoCAD.

So in the bat file, it actually gives you options to uninstall AutoCAD. Why is that a bad thing? Because you could have AutoCAD, you could have Civil 3D, you can have AutoCAD mechanical, MEP, PNIC, I mean, you can have all these on there. So as Jerry mentioned, if you uninstall this, you potentially just broke your other products. So be fully aware of what's actually in this file.

The greenish color is the core application. So this is kind of your safety. This is your Civil 3D. So it's the Civil 3 core application, the language pack. Down in here is another Civil 3D language pack. The object enabler. Maybe you don't want to uninstall that, so you might take that out or not.

So those are kind of your safeties and all in the green. The blue are shared components that gets installed with most of the applications. So, for instance, like if you did a find and replace colon, colon, and replaced it, you've just uninstalled the Autodesk license service. So that's not a good thing.

You actually can come in here-- there's some of them that's manually uninstalls only. But application manager, A360 Desktop. A360 Desktop actually runs All the Time Resident. So one thing I found is when I uninstall A360 Desktop, sometimes my windows environment starts acting up on me. So I'll actually do a preliminary process of actually killing that if I want to uninstall it. Or just leave it alone. Don't uninstall it.

Also our other components. So basically, these are again shared applications. Or it could be anything from your material libraries. It could be anything like that. In this case, we also have AutoCAD Architecture. We have the language pack for AutoCAD Architecture. We have AutoCAD Map 3D.

So as you can see, this file could be kind of dangerous if you just did that find and replace. So be fully aware of what you're actually going to uninstall, you. Don't mess up your desktops. Now, basically this is just really-- uninstall the application, the components that you actually want to uninstall.

One thing that's not in the .txt file is actually, when you go through your Deployment Wizard, if you actually go through the update process and you apply to HotFix or you apply to service pack or something like that, it's not going to be in the .txt file. So it's unaware of those secondary processes.

Also if you have any kind of a secondary installer that runs through your wizard, it's not going to know about those. So just be aware that.

The components, just basically make sure you are uninstalling what you want. And then, like I said, some of these shared applications or components may be running on the PC at the time. And if you kill a process-- if you actually uninstall while it's running, it can make your Windows Explorer unstable.

And then once you do it, test. Test, test, test, test. I've done this every year. 2017, I install all my 2017 products. I run the uninstall bat files and stuff like that, and test it. And that's where I run into, OK, I lost my Explorer. I need to go check what's going on. I accidentally took one off or something like that. So test it.

On this-- as far as the scripts, once you get these bat files created, then you can just basically use any technique that you're currently using to push the software, push the bat file also. If you're using SCCM on the push deployments, the bat file is in the deployment. So in your SSCM advertisement, this is the install for AutoCAD 2017. And here's the uninstall for 2017.

It's the same package, it's just running the bat file. It's going to need all those files and stuff like that, the same as the installer does. I use a little different technique myself. But the goal of the bat file, or the benefit of the bat file is, how many times have you installed a product, need to uninstall it, and then try to do it through programs and feature. You have to go in, OK, well, this was Civil 3D. Wait, here's all these other components that got installed with Civil 3D.

That's what the benefit of the bat file is. It tells you every component that got installed. And you can actually easily uninstall those quickly.

Jerry's like, He's calling me now, I guess, the GUI master or something like that. I like GUIs for my end users, so I've developed these tools, not just for me, but I give them to all our PC administrators and all. And all, so for the bat files. I actually have an option. This is a tool that I wrote to install software, update software, and do everything.

So they can actually come in here and select any application, dump it, and then choose to say, it's going to do an uninstall, and then run it. And it will actually uninstall all the applications for them. And all it's doing is launching the uninstall bat file. That's all it's doing.

So let's talk about the MSI switches for deployments. The one thing about Deployment Wizard that happens is it goes through the creation process, and it creates you shortcuts. And these shortcuts are simply like I created 64ACD17. That's the installer shortcut.

It also has a create and modify deployment shortcut. It also has an update, that image shortcut. So in a larger environment, what happens when you start sharing these out? These deployment images with Windows shortcuts? They're no good anymore.

Windows shortcuts are tied to the source. Once you start copying these packages all over the place, source is no good. And you're not going to copy it at 20 different locations, and then go edit the INI, or edit the shortcut, right? Because you will now know, and you should know, as far as the parameters that they shortcut are actually doing.

So this shortcut that actually gets created is, in the Deployment Wizard, you have the option of saying, run this silently or not. So that's a checkbox in the early images. So if you chose it to run it quiet or silent, it's actually pushing the QBI.

I actually, from Legacy working with Jerry, there used to be a time when the 4/W was very important because it basically told the deployment, forget all the paths that the original source came from. This is where the package exists. And also the 4/W helped to eliminate in the INI files where it still pointed to the original source, that 4/W will ignore that.

I also have times where I need-- if you chose not to run it silently, you're going to get this 4/I just to install the package. I still use the W anytime moving packages. But that's good for troubleshooting and having your users go in and report an error. If you run it silently, and it doesn't show up, rung it nonsilently, and then you can actually see the error message. Silent-- silent install, whether it's successful or not, it's going to be silent.

Modify. The 4/MD. That's actually will modify this deployment, and it's going to be that specific

deployment image. So you don't have to rerun that shortcut. I did so well up until that last moment. That shortcut here, because this is broke now, because you copied it over. And if you sent it to another location, and they need to modify it to add their content to it, all you have to do is pass that MD.

The update. We all might know we're going to run the appli-- we can run the application manager on the desktop. It's going to go out and see that you got all these products installed on your desktop. And it's going to say, OK, there is a hotfix for AutoCAD. There's this service pack for Civil 3D. And so forth.

Well, if you actually run this command, it's going to say, it's only going to give you the options that fit to that deployment. So same as the Deployment Wizard, when you went through and you finished creating your deployment, and the last option was to do an update, and it only showed you the updates that were valid for the applications that you chose to include in the deployment, that is how you get that. So basically, this shortcut that's in the tools folder is this command line.

We talked about the uninstall bat. And again, why know these? Because shortcuts break and all. And kind of going back to my GUI tool that I sent to my PC administrators and stuff, as I give them that GUI, they can select any application they want, and they can choose, I'm going to do a batch install of 10 pieces of software. So I'm going to choose batch install, and I'm going to choose all 10, tell it to go, and be done, and not worry about it.

Or I'm actually going to come in here. I need to modify the five that I have, because I either need to add some content to it or there's a service pack outside. So I'm going to add these five, and I'm going to run to modify, and it's actually going to go through. And I'll jump to the next slide. I'm going to run the modify on this package, and it's going to pop back that same kind of wizard that you had when you run that shortcut.

And the same thing with update. It's focused on that image. All right, I think we got through that. And I think we have enough time for Q&A.

PRESENTER 1: Who we go? We've got a few minutes for some questions. Yes, sir?

AUDIENCE: [INAUDIBLE]

PRESENTER 1: OK, so the question was, why is-- you think you pushed out a new environment variable, and

you did.

AUDIENCE: [INAUDIBLE]

PRESENTER 1: Sorry?

AUDIENCE: [INAUDIBLE]

PRESENTER 1: It's finding a breadcrumb, almost for sure. Look under HKey, current user, HKey, local machine, software. And below there, look for a value ADFLEX_License_file, I bet you find one.

AUDIENCE: [INAUDIBLE]

PRESENTER 1: Delete them. Yep.

AUDIENCE: [INAUDIBLE]

PRESENTER 1: [INAUDIBLE] at LEC is, if you have a [INAUDIBLE] at LEC, that's the last place the application is going to look. So it doesn't usually cause-- that's not the cause of slow startup. OK?

AUDIENCE: [INAUDIBLE]

PRESENTER 1: Sorry?

AUDIENCE: [INAUDIBLE]

PRESENTER 1: Honestly, I don't run into that, honestly. So if-- anything's possible. You know, there are always exceptions. But I haven't seen it.

AUDIENCE: [INAUDIBLE]

PRESENTER 1: I'm sorry?

AUDIENCE: [INAUDIBLE]

PRESENTER 2: Are you installing from the deployment and it's actually looking for a source that doesn't exist?

AUDIENCE: [INAUDIBLE]

PRESENTER 2: I would try the /W and try to get that. Because /W is made to really-- cause the INI, if you look at the INI file of the deployment, there could be paths in there that are still related and all. So you might want to try that.

AUDIENCE: [INAUDIBLE]

PRESENTER 2: Yeah, and just ping, and you can ping us if you want to follow up on that question and stuff, and say--

PRESENTER 1: Anybody else?

PRESENTER 2: I think the gentleman here.

AUDIENCE: [INAUDIBLE]

PRESENTER 2: That's a really good question. The question was, do I have to sync the version of lmutil with the version of the daemons on the license server. And the answer is, kind of, sort of. OK? It usually works just fine if you didn't. But there is a possibility that if you're using an old version of lmutil, older than the servers, you might get unpredictable results because it doesn't understand everything.

I know, for instance, the vendor daemons from Autodesk, as we go through different features, and that's why we require that you upgrade your license manager, because we use a new functionality available. And if you use anything from the old thing, we're not going to pull that information with that version of lmutil.

So it's best to always use the latest version of everything whenever you touch it. That's the best advice I can give you. And the worst part about it, is it'll seem to work. But it may not be completely correct if you're not up to speed on it.

Anything else? Anybody? OK, hey, listen, thank you, everybody. I really appreciate you. Tiny and I both appreciate you coming and spending the afternoon with us. And we'd like you to take a moment and do the survey and give us some feedback. We'd really appreciate it. Thank you.