Nicola Tavares: Welcome to the ITS Standards Training.

Ken Leonard: ITS Standards can make your life easier. Your procurements will go more smoothly and you'll encourage competition but only if you know how to write them into your specifications and test them. This module is one in a series that covers practical applications for acquiring and testing standards based ITS systems. I am Ken Leonard, the director of the U.S. Department of Transportation's Intelligent Transportation Systems Joint Program office. Welcome to our ITS Standards training program. We're pleased to be working with our partner, the Institute of Transportation Engineers to deliver this approach to training that combines web based modules with instructor interaction to bring the latest in ITS learning to busy professionals like yourself. This combined approach allows interested professionals to schedule training at your convenience without the need to travel. After you complete this training we hope that you'll tell your colleagues and customers about the latest ITS Standards and encourage them to take advantage of these training modules as well as archived webinars. ITS Standards training is one of the first offerings of our updated professional capacity training program. Through the PCB program we prepare professionals to adopt proven and emerging ITS technologies that will make surface transportation safer, smarter and greener. You can find information on additional modules and training programs on our website www.pcb.its.dot.gov. Please help us make even more improvements to our training modules through the evaluation process. We look forward to hearing your comments and thank you, again, for participating and we hope you find this module helpful.

Nicola Tavares: Throughout the presentation this activity slide will appear indicating there is a multiple choice pop quiz following this slide. The presentation lecturer will pause at each quiz section to allow you to use your computer mouse to select your answer. Selecting the submit button will record your answer and the clear button will remove your answer if you wish to select another answer. You'll receive instant feedback on your answer choice. Please help us make even more improvements to our training modules by completing the post course feedback form. Today's webinar or module is A315b Understanding Requirements for Actuated Traffic Signal Controllers based on NTCIP 1202 Standard part 1 of 2. Your instructor Ken Vaughn received his bachelor's degree from Tulane University and his master's degree from Texas AMU. He coordinated the traffic signals in Los Angeles County in the early nineties and his graduate research study, the performance of actuated signals in coordinated systems. In 1994 he became involved in ITS Standards and was a founding member of the NTCIP Joint Committee in 1995.

Ken Vaughn: Hi, I am Ken Vaughn. And we're going to talk today a little bit about the A315b writing requirements for traffic signal controllers. And the target audience today is

the traffic management and engineering staff. These are the people that will be writing the specifications and being responsible for the management of the new system. TMC and operations staff, of course, these are people that are going to be managing or operating the new system. Traffic signal maintenance staff, of course, these will be the people that will have to maintain the new system once it's deployed. System developers that are responsible for developing the system and private and public sector users including the manufacturers who have to build these specifications. So I hope you're in one of those groups. And also hopefully you've all ready taking some of the prerequisites. There are several for this course starting out with some introductory courses I101 Using ITS Standards, an overview and A101 Introduction to Acquiring Standards-Based ITS Systems. Those are followed on with some courses that related to user needs development. A102 Introduction to User Needs Identification. A201 Details on Acquiring Standards Based ITS Systems and A202 Identifying in Writing User Needs When ITS Standards do not have SC Content because, of course, the 1202 standard for signal controllers do not have that content currently. Those courses are followed on by some requirement level courses. So A103 Introduction to ITS Standards Requirements development and A203 Writing Requirements When ITS Standards do not have SC Content. That's followed up with some communication module and that's C101 Introductions to Communication Protocols and their uses within ITS applications. Finally, that comes up with a module dedicated to traffic signals, in this case, the user needs for traffic signals A315a understanding user needs for actuate traffic signal controllers based on the NTCIP 1202 standard. All of those line up one after another as introductory courses to this particular course A315b Understanding Requirements for ASC based on NTCIP 1202 standard. Now, this particular course is actually in two parts. This is part one of two parts. There will be a second part dealing with some more detailed issues related to signal controllers. And then finally, you may also be interested in taking T315 which is applying your test plan to NTCIP 1202 ASC Standard. So that's the basic curriculum path you're on so you're almost done. The learning objectives for this course include learn how to develop requirements using the NTCIP 1202 version 2 standard. Achieve interoperability and interchangeability which, of course, is very important in the goal of NTCIP. Understanding traceability and how that plays into achieving interoperability and interchangeability and how that really comes into play in testing. And then learning objective number four developing the specification, seeing how all of this material goes into that final product. Now, we did mention before that there's a second part to this module that goes on to talk about managing special issues for ASC and also incorporating requirements not supported by the standardized objects. So both of those will be in part two of the module. This part one will deal with the first four learning objectives listed here.

Ken Vaughn: With that, the first learning objective we'll cover a number of issues including reviewing the structure of the standard so you understand how it's laid out. And then learning how to identify requirements from various different sources including user

needs, SEP based standards, conformance groups and just configuration control and monitoring perspectives looking at it objectively and deriving those requirements. We'll also discuss some of the criteria that we've established for well written requirements and then we'll develop some sample requirements based on our previous module A315a. So with that we'll talk about the structure of the standard a little bit. We've mentioned this before that section one starts out with the general section and then it immediately dives in to the details, the design details. Section two talks about object definitions and section three lock object definitions, which is basically a more efficient, bandwidth efficient way of exchanging data. But those are design level objects. What's missing from this particular standard is the SEP content and we'll talk about that in a second. Annex A goes on and defines an information profile including your conformance groups. And this basically is what identifies and takes all of the different objects in sections two and three and puts them into logical groupings so you can quickly identify which particular sets of objects are particularly included and the ranges you need on those objects. Annex B defines these set of consistency checks. As you're probably familiar, signal controllers are complex devices that are safety critical. And as a result, there are a number of consistency checks defined so that when you download information, the new timing parameters to your signal controller, they will run through these consistency checks to make sure that everything seems to be appropriate before implementing those new parameters. Annex C defines a concept of operations but this isn't really a concept of operations like you would normally think of in a systems engineering context. It's really more of a set of design templates used to describe how we change your information within the device. And then finally Annex D talks about deprecated objects. And these are objects that were previously defined in earlier versions of the standard, but for one reason or another, we've decided that they're no longer the best way of doing things. So they're kind of legacy information that if you're dealing with older equipment you may need to be knowledgeable of these items. But with new designs we're saying don't use these. Use the newer design that's in the body of the standard. As we mentioned what's missing from the standard is the systems engineering content. That includes user needs. That was discussed in the previous module, A315a and then also the requirements, dialogs and traceability. And those three concepts will be discussed within this module. So how do we define requirements and identify them? There's various sources as we discussed. The first is looking at that previous module, we defined our user needs. So if you're looking at a top down approach that's how you follow the systems engineering process. first you define your user needs. From those user needs you analyze them and develop your requirements. So that's one approach. The second approach is to recognize that there are several other NTCIP standards available and they've all ready gone through this process. Now, those are different devices and by and large have different needs, but sometimes your needs will overlap. So much more generic sort of needs may be addressed by some of these other standards and it's worth investigating that. And, in fact, we've all ready done a lot of that work for you and that's recorded in your student supplement. So we strongly encourage you to download that student supplement,

reference that when you're developing these requirements because there's all ready some text in there that will be very valuable for you. A third source is the conformance groups. So whereas the user needs is a top down approach you can also look at the design all ready defined within the standard in the conformance groups and objects and look at it from a bottom up approach, kind of a reverse engineering approach. We know what the design is. Why would we need that design? And what you can do is kind of reverse engineer not saying that you necessarily need everything that's in the standard but by looking at those design level details you may discover that there's things in there that you hadn't thought of. So it's a really good second check to make sure you've thought of everything that you may need to include within your specification. And then finally, just using really good engineering judgment of looking at the problem from several different perspectives including configuration perspective, control perspective and the monitoring perspective. And we'll talk about this. This is really a very iterative sort of process. And, in fact, as we get right into it the user needs when you're developing them from user needs it is a top down approach and it's a recursive iterative process of discovery. And you just keep asking the questions, what do I need in order to achieve this? What do I need in order to achieve my user need? And as you ask those questions you discover that maybe your user need isn't well written enough. So you need to go back and update your user need to make sure you capture the whole example. And then we'll give you an example of that here.

Ken Vaughn: So if you remember from the previous module we had previously said that we would take the controlled selection of timing pattern user need here that you see in 2.1.3.1 and that we would within this module we would detail that and that's what we'll do right now. So that user need was stated as the agency needs to be able to control intersection timing to accommodate the demands on the signal, while also providing green waves to allow smooth and efficient flow of traffic through the signal system. Well, there's a few things when you really start analyzing that statement, there's a few questions come to mind. One is well I need to configure the timing pattern because I need to be able to control intersection timing so I need to configure timing pattern. What it doesn't say is whether this is a static timing pattern or if I need to change it. And if I need to change it periodically how do I change it? Is it by time of day? How many timing patterns do I need? How do I select the timing pattern time of day? Is it manual command? Is it automated somehow? And then once I change it, can I monitor it? So you see that we've gone through this process of I want to control selection of timing pattern but within that control logic, let's look at it from a configuration perspective and then from a control perspective and then from a monitor perspective. From a configuration perspective, how do I time it? What do I need to time? How many of these do I need to time? From a control perspective, how is this going to be controlled? Is it manual or is it some automated system? And then from a monitor perspective, how do I monitor what's being controlled? So what's the current timing pattern? Who controls it? Questions like this. So from those questions we can go back and revise our user need statement and

that we say well rather than just controlling it we're actually adjusting it dynamically based on the dynamic demands. And then we also add a whole other paragraph describing it further so that the pattern will typically be selected from a pre-defined list by the central system. So it's going to be externally controlled based on network conditions but the controller needs to be able to default to a specified schedule if any problems occur with receiving these commands. And field personnel should be able to override these commands. So we have identified at least three distinct states here that one is externally controlled, one is controlled by a schedule and one is controlled by local personnel. So we need a way to manage all of those. The controllers should allow for storage of sufficient patterns for all of these purposes. Now, notice here we're talking about a user need. So we don't necessarily have to be as specific to cite a precise number. But the time it gets to requirements we will need to but not necessarily in your user needs, unless that's really, really critical from a user needs perspective. Once you get to the requirements, though, then you do start getting very, very specific. Not only do we need to configure a timing pattern, but now we need a measurable item. We need to support at least 32 timing patterns. We need to configure the timing pattern selection logic. Is it going to be local control, central control or schedule control? We need to be able to define how you activate the timing pattern remotely, how you activate it per schedule and how you override it locally. And then finally, if you're going to have a schedule, well clearly we need to support a schedule as well. Now, you'll notice that these are not necessarily your requirements per se. These are simply identified requirements. At some point, later in this presentation, we'll start talking about how you write these up into precise, clearly stated requirements. But right now, we're just identifying them and you'll see why here in the next slide that we don't want to reinvent the wheel. If some of these requirements are all ready included in other standards, maybe we'll just reference those and save us some work. So right now, the first step is just to identity requirements, later on we will go ahead and make sure that we have the text for those requirements. But getting into that there are other SEP based standards. And you may remember from the previous module that some of our user needs, in fact, were repeated from the previous standard and that's a prime example of where there's probably going to be a lot of overlap in your requirements. If you had a previous user need from another standard, you're likely to be able to take that whole user need and set of requirements lock, stock and barrel and use them in this standard as well and this procurement as well. An example of that is one of the user needs that we referenced in the last module was live data exchange. And virtually all of those requirements related to live data exchange would also apply to traffic signals. We'd also typically reuse the associated requirements. Some requirements, though, can be copied for slightly different uses. So if you think, for example, supporting a schedule, well say a message sign needs a schedule in order to control the display of a message. The user need is to display a message, not to control signal timing. So the need is different but the fact that they both use schedules is similar. And, in fact, you find out that both standards reference the same mechanisms, the same objects to control that schedule. And as such you should be able to just copy virtually all of those requirements related to

defining a schedule, and monitoring a schedule and everything over to the signals and replicate them. So signals seen as schedule for timing pattern, message sign seen for message display, but minor some tweaks you should be able to copy much of that information. And once again, we've done a lot of this for you and that's contained in the student supplement.

Ken Vaughn: The next item is so identifying requirements for your signal controller, in particular. Consider if similar requirements exist elsewhere so that this is how you go about identifying those requirements that may exist in other standards. The first thing you do is you identify what your standard needs to do, what your device needs to do. And then you kind of think broader, can some of these things that I need to do be implemented by other standards? Do they have similar sorts of needs or design patterns that I can reuse? And if so, then you pick up one of those standards and you glance through it and see if they have something that you can, in fact, reuse. In particular, you can go immediately to the protocol requirements list and that's where you can quickly review all of the names of all of the requirements to see if something sounds like it may apply to you. If it is, then you can reference it. We want to emphasize the fact that by and large you want to reference requirement and not copy them per se. Even if you modify the text a little bit you still want to make sure you keep a reference there so that any reader is aware of the fact that you're repeating something that's all ready been used in another context. This allows a developer to realize that hey maybe I all ready have codes to implement this. I don't have to reinvent the wheel myself. So it promotes reusability by making sure that you reference where you're grabbing this information from. And, once again, we actually have done a lot of this work for you for signal controllers within the student supplement. So as an example of this log data exchange is another user need that was copied over in the user need section of the last module. And virtually all of these requirements will apply to signal controllers, just as they applied to the message signs that we referenced in the previous module. So, for example, set time we need to do that as well. Set time zone we need to do that as well. So despite the fact that these are all from the message sign we would include these within our procurement as well. Now, if it's an example from something that we're only importing requirements and not the parent user need is, once again, the support sets supporting a schedule. In this case, supporting a schedule includes retrieving a schedule, defining a schedule, setting a time, setting a time zone on down the line. What you'll notice, though, is if you actually go in and read, retrieve a schedule or define a schedule within the DMS standard it references everything in relation to defining a schedule for a message. So we need to go in and tweak that text a little bit to make sure that it's related to setting a schedule for timing patterns. But other than that, we can largely reuse that requirement. So a lot of that text is all ready there. Once again, we've all ready updated all of this for you. One note you'll notice on both of these slides the set daylight savings mode was highlighted in green. The reason we did that is because since the CMS standard was approved, the daylight savings mode mechanism was changed. You may remember a few years ago, Congress changed the rules for

daylight savings time. As a result, we had to modify our standards a little bit. We actually ended up modifying it so that we're now completely flexible. We can handle any sort of change they may do in the future, but that was different than what we had before. So you can't just immediately copy over what they did there. You will have to make some refinements of that particular feature. But everything else will fall straight through. The next thing is we talked about the conformance groups earlier. And conformance groups you may remember are the bottoms up approach. So it's kind of reverse engineering bottoms up approach of taking the design that's in the standard and saying is there anything the we're missing that we haven't included for in our user needs at sign in? The way you go about this is by looking at the particular conformance groups. And so you think of a particular category of say coordination and you look at the coordination conformance group and you look at each object in that conformance group. And you consider, are any of these objects that I haven't really thought of yet in my user needs. Some of them you may have thought about and said well we don't need that. Some of them, though, you may not have thought about it at all and it may be worth investigating to see if you need them. There may also be some that you don't intuitively understand. You may have to say well, what exactly are they talking about there? The advantage is in this conformed scripts table, as we'll show you, it includes a reference to the exact clause number where that object is defined. So it's very easy to go from the conformance group back to the exact clause number and get a precise definition when you need it. You investigate those particular objects further so the ones you don't immediately understand or haven't thought of yet, you investigate them further. And then determine if that functionality is needed for your particular project. And if it is, then you need to go back and revise your user documentation, user need documentation to make sure that explains why you're including this feature because, once again, everything should be traceable so that everyone knows why things are included. As an example, you may be looking down at the coordination conformance group and you get to coordCorrectionMode. And at first, you may not recognize what that means but then you start looking at dwell, short way, add only and you realize, this is how the controller adjusts from one timing pattern to another timing pattern when a change is made. And there's various different timing logic you can use to make that transition dwell, short way, add only. We won't go into the details of those but those are all different options. Now, perhaps you don't care, in which case you don't need to call out this particular object. But if you do care and you want to be able to vary the status within the controller then you would need to support this object. If you care, but you always want every signal to be say dwell, then you don't really need to worry about this object. You just need to specify that you want your signal controller to support dwell. You make sure that's how they're delivered. But if you want to be able to modify the setting you need the support of this object. So this is where you would say, okay, we now need to add a requirement. We now need to go back to our user need, make sure it's covered within the scope of that user need.

Ken Vaughn: So we'll identify yet one more requirement to add to our list, configure timing pattern transition mechanism. And then we will move on to the next approach which is identifying our requirements from the three different perspective configure, control and monitor. And this does highly relate to the top down approach of just constantly asking questions. But we're really asking these questions all of the time. And so every time you're even looking at a control requirement, you're always asking how am I going to configure my device to handle this type of control? How am I going to monitor a device in relation to this control operation? So what you may discover is that you may end up with some overlaps of requirements during this process because some of your user needs are going to be monitor based. And now you're going to be developing monitor requirements related to your control items as well so you're likely to end up with some overlap. That's fine at this point in the game. Don't worry about that. The key here is making sure you identify all of your requirements. Once you've identified all of your requirements here done to that process, then you can go back and refine your traceability. Does this really fit here? Or is it really better somewhere else? And if it's better somewhere else maybe it applies in both places. But the key is making sure you get all of your requirements defined and the traceability then comes after that. So as an example, something like controlled selection of timing pattern, I can do a number of things to monitor that control operation. So I can monitor the pattern configuration. I can monitor the current timing pattern. So what have I controlled or selected? Monitor the last timing pattern requested so that may be different than what's actually operating. I maybe want to know that. Also monitor the source of the last timing pattern. Monitor the transition mechanism selected, monitor a timing pattern schedule. So all of these things are monitor operations on a control type user need. Once again, some of these may overlap. You can worry about that, later. So through this process, we've identified a total of 24 potential requirements for the controlled selection of timing pattern user need. They could be easily restated to be more or less so we're not going to get into that. The main thing we want to emphasize here is for most of your user needs, you're going to end up with multiple requirements. And when we say multiple it's quite a few, really, requirements will relate to each user need you identify. And it's very important that each requirement traces back to at least one user need, otherwise, you end up with a question of why am I requiring this? If there is no need, why am I requiring this? So every requirement should trace back to a user need. If it doesn't it means maybe this isn't a requirement I should be requiring. Also only use requirements that apply to your project. It's very, very tempting to just grab someone else's spec and use it lock, stock and barrel but recognize every single requirement increases costs. It increases costs related to the development of specification, of it being reviewed over and over again through your process. It's going to increase the cost for manufacturers to get that specification, review it, identify how much it's going to cost for them to bid. Also, once the successful bidder is selected, it's going to increase the cost of the product. It's going to increase the cost of testing that product once it's delivered. And that whole cycle repeats when you go for your next system upgrade, you go for your next procurement. So really if you can minimize the number of

requirements you have you save everyone money in the long run. At that same point, you don't want to go too short. You want to make sure you have a complete specification with everything you need. If you don't have a complete specification there's various problems that occur. The delivered system may still be conformance with the standard but it may not include the optional features that you need on your project. Another issue is it may be fully conformant with your specification but if it doesn't meet your project needs, you still have a problem. Further, if you really have an incomplete specification you may get features that are non-conformant. They may do what you wanted it to do in your functional spec but in your interface specification they did it a different way which is not conformance to the standard. So, once again, that doesn't meet your needs either. The bottom line is you may end up getting an incomplete system that only partially implements the functionality you need. So preparing a complete specification is very important. So once you've identified all of your different requirements, the next task is to actually write them up. And writing them up you need to make sure that they're properly stated. They need to be following a simple example to make sure you have concise requirements. So we've developed this basic sentence flow of actor, action, target, constraint, localization. The actor is who or what does the action, who initializes the action. The action is what is to happen, what is that actor required to do? And then the target is why is that action being acted upon? And then constraint is an optional feature that identifies how to measure success or failure, constrains that action. And finally, the localization is identify circumstances under which the requirement applies. And you'll notice, we'll get into here in a second, sometimes if you have both the constraint and localization present, sometimes you need to move the localization to the start of the sentence to prevent ambiguities and we'll talk about that in a future slide. But to give you an example of this actor versus target within the NTCIP environment, most of our requirements are going to be written from as the central system being the actor and the ASC being the target. That's because the NTICP primarily uses the request, response approach. So the actor is a central system. It initializes the request to perform some action upon the ASC. The ASC then simply follows the design details of the SNMP protocol that has a particular piece of information as request had said. It's required to respond by the protocol design details. And then that response comes back to the central system. So in the vast majority of our requirements you'll see the central system being the actor and the ASC being a target. The other thing that you need to keep in mind is the writing requirements is that every requirement needs to be kept concise. Everything about it needs to be necessary. It needs to be attainable so you can't say well just do it in one millisecond. You have to have attainable specifications. And it needs to be a standalone requirement. You really don't want to have your requirements start referencing other requirements. It really complicates things and that's exactly how you get problems in interpretations. It needs to be consistent with all other portions of your specification. It needs to be, of course, unambiguous and then finally verifiable. Once you get a product delivered you need to be able to test it to say either yes it does meet this requirement or no it does not. If it meets engineers' approval or something that's not very verifiable. It becomes very subject and they're very

hard to hold the manufacturer to that sort of requirement. But as a simple example to start out what, this is a requirement that we imported from the DMS standard NTCIP 1203. The central system shall configure the ASC with the current coordinated universal time to the nearest second. So you parse out the sentence, the central system, well that's the actor. And then shall configure well that is the action that is being taken. The ASC is the target with the current coordinated universal time to the nearest second is a constraint. That's the information being configured within the ASC. So those are the different parts of that sort of requirement. Pretty straight forward example. Now, we'll develop our own example using our own sample requirement. In this case, it's going to start out exactly the same. The central system shall monitor, in this case, rather than configure the ASC to determine which timing pattern is currently active. So once again, the actor is the central system, shall monitor this time is the action. The target once again is the ASC. And the constraint of time is to determine which timing pattern is currently active, so a fairly common process. But as I mentioned before, if you have both a constraint and a localization, you may need to separate these out. And this next slide explains that a little bit. If you just go through and you look at the sentence as the different parts it's the same basic structure. It's the central system shall configure the ASC. And then the constraint is time, with timing pattern information subject to ASC imposed validation rules. Okay, that sounds good. Localization when requested by the operator. Well, okay, that's good. The problem is if you put this localization immediately after the constrain, you end up with an ambiguity. The ambiguity is that localization could either apply to the sentence as a whole and really to the shall configure when requested by the operator, the central system shall configure. Or it could be interpreted as a local modifier clause to the previous constraint. In other words, it could be interpreted as the timing pattern information subject to ASC post validation rules when requested by the operator. So those rules would only apply when requested by the operator. Well, that's not what we mean. What we mean is the central system shall configure when requested by the operator. So what we do is we move this last localization clause to the front of the sentence. When requested by the operator, the central system shall configure the ASC with timing pattern information. You structure it that way. And that prevents that sort of ambiguity from creeping into your specification. You'll also hear that the term pattern is shown in green. That's just t indicate that this is a defined term within NTCIP 1202. And it's good to always go through your requirements looking for any particularly different sort of terms that may be used, custom to your particular context. And make sure that those are defined so that there's no ambiguity created by those terms. Well, that concludes learning objective number one. And will bring us to our first quiz. Which of the following statements is not true? So go ahead and answer. Option A, user needs are used in a top down approach to identify requirements. Answer B, you should read every SEP based standard in order to get ideas for requirements. Option C, conformance groups are used in the bottom up approach to identify requirements. Or option D, you may discover overlaps in requirements from different user needs. Once again, which one of those statements is not true or is false. Which one of those is not true, A, B, C, or D? And go ahead and make your selection.

Well, that's great and hopefully most of you answered B is the one that is false. You don't need to read every SEP based standard to get ideas. You should only reference those standards when you think they may be of benefit. And, in fact, you really only need to look at your student supplement. That should identify most of the ones you need for signal controllers. But if you're doing another device, you could look at other standards if you see there's a benefit. The other answers are true. User needs are used in a top down approach. That's the traditional systems engineering approach. Likewise, conformance groups are using a bottom up approach. That's kind of a reverse engineering approach but you can also benefit from that sort of solution. And then finally, there may be overlaps and requirements but harmonization of this issue is a later topic. So that does conclude our learning objective number one. We talked a little bit about the structure of the standard. We identified rules for identifying requirements and identified a number of requirements for the sample. And then we provided some sample actual text of requirements. We provided one from a previous SEP based document. And two sample requirements that are specific to our particular standard.

Ken Vaughn: That does bring us now to learning objective number two, achieving interoperability and interchangeability. The first thing we'll do is we'll review the definitions of those terms. And then we'll move on and talk about some interface dialogs particularly related to SNMP because that's really how you achieve interoperability and interchangeability. We'll also talk about NTCIP objects a little because that's kind of the heart of the standard and how NTCIP 1202 traces over to NTCIP 1201. We'll also talk about sample dialogs and how those are written and developed and then we'll prepare some sample specification text for those dialogs. So interoperability is a term that is standardized within the software industry. And it is defined as the ability of two or more systems or components to exchange particular information and to use the information that has been exchanged. So not only is it exchanging data between two systems, but it's allowing those systems to understand that information and to use it. That's what interoperability is. Interchangeability, slightly different, it's not actually a standardized term with the industry but it is a term that has been defined by the U.S. Department of Commerce. They define it as a condition which exists when two or more items possessed such functional and physical characteristics as to be equivalent in performance and/or durability and are capable are being exchanged one for the other without alteration of the items themselves, or adjoining items except for adjustment and without selection of fit and performance. Now, obviously within NTCIP we don't do a lot related to specifying the physical characteristics of our devices. So from our perspective, the interchangeability really focuses on functional aspects, not so much the physical characteristics. Other than that, that definition applies. Well, that brings us to how do we achieve interoperability and interchangeability? And the way we do that is through defining precise dialogs. By defining precise dialogs we're able to really clarify exactly how these systems interchange data and then make sure with the standardized definition of terms that they can use that information. And if they all do it the same way, then you get interchangeability as well. So

there's two basic protocols used within the NTCIP for these types of exchanges. The simple network management protocol and the simple transportation management protocol. STMP is really just a customization of SNMP. SNMP is a very generic Internet standard used probably on the modem on your computer or on your Internet connection. SNMP is very, very common standard. We've adopted it within the ITS community for our needs. But we've also recognized that it is rather bandwidth intensive. And particularly for signals at times you need something that's more bandwidth efficient. And we developed the STMP as an alternative that does the same basic features of SNMP. It just does it more bandwidth efficient manner. So we'll talk about that more in part two of this module. This module will talk about SNMP. We'll talk about the messages involved within SNMP. And then we'll get into the objects and object ranges. For SNMP there are three major dialogs involved. The first is get request. It involves a get message which automatically generates from the device a response message. Each of these messages it's a very generic sort of message that contains the list of information that you need from the device. So I'm going to get a particular set of objects and then the response from the device is going to be the response message with that same list of objects, this time included with our values. The second dialog is some very, very similar. It is a set dialog that sets a list of objects, this time with the values you want them set to. And coming back from the device is a response containing that same list of objects and the value that was set to them. And then finally is the get-next. This is a little bit more explanation is needed. Basically, this is very useful if you have a table and you don't necessarily know how many rows are currently in that table. I can go one after the other, okay, give me the next row in the table, give me the next row in the table. And when it jumps to a different column in the table or jumps beyond the table you know that you've all ready retrieved all of your information from that table. So the get-next is a very similar to the get request once again. The get-next request containing the object list, response will come back with a response message containing the next object rather than the object specific instance you referenced. So those are the three basic dialogs. As you can see, a very basic sort of information, once you find out as we go through all of our other dialogs that we are going to define is made up of combinations of these three dialogs. So the next step is once you know the basics of that dialog, well, how I fill in that object list? How do I know which objects to retrieve? I mean that's where the conformance groups come into play. Annex A2 of the standard NTCIP 1202 version 2 lists the conformance groups and you can figure out what sort of category of information do I need if I'm interested in coordination type data, when I look at the coordination conformance group? Each conformance group identifies a list of objects, a name that identifies a function and a clause number where you can find out more information about that object. So if you look at the name and a lot of times the names will be self-explanatory. At times you'll say well I'm not sure exactly what that means, let look over at that clause and I get the full definition. The object is, in fact, defined in a computer readable format and we'll discuss that here in a second. So this is what the conformance group table looks like. And as you can see, if we're interested in monitoring the current timing pattern, when you look down well we know a

current timing pattern that's going to probably be under coordination conformance group. And we look down under coordination conformance group and we find coordPatternStatus. So that's probably the object that is going to define which pattern is running right now. And to verify that we can glance over at 2.5.10 as the clause number. We jump to that portion of the standard and we find this. This is the computer readable format of that object. It defines that this is, in fact, an object. It is going to be representative as an integer from 0 to 255. For your reference that happens to be one byte of information. It's access is read only which is what you'd expect from a status sort of object. You can pretty well ignore the status object of optional. That's kind of meaningless. It's an artifact of how we have to define things according to the Internet. And actually they have kind of replaced that. So it's kind of a meaningless descriptor. What you're primarily interested in is the description which starts off with the definition field. This object defines the running coordination pattern mode in the device, which is exactly what we're looking for to monitor the status of the device. So you'll see that it has the value of 1-253; 253 is the pattern number; 254 is free operation; 255 is flash operation. But you'll also notice down at the bottom is this is strictly for the computer to understand. We would call this object coordination pattern status. A computer, when they exchange it over the wire, they essentially are calling it coord10. That's just their identifier of how to identify this particular object.

Ken Vaughn: So now that you've found your object you want to exchange in order to monitor the current timing pattern, now you can create your dialog. And that dialog looks like as shown in this figure. It's a simple get of the particular object we're interested in coordPatternStatus.0. And the agent will receive that and will send back the response. We don't really need to label that response. That's kind of an automated return. That's defined within the protocol itself. Now, you may be asking yourself, why do we add this little dot-zero to the end? Well, some objects are what we call scaler objects. There's only one instance of the object ever in the device. And in that case they get a dot-zero. If the object can be contained within the table, then they need to have their index added. So it may be the first row of the table, the second row of the table, the third row of the table on down the line. And, in fact, in some cases you may have W-indexed items and we'll talk about that in a second. But in this case we're a simple object. It's a scaler object. And there's only one in the device. So it's coordPatternStatus.0. We're going to get that object. Now, that's the diagram of what we're trying to do but we need the text because the text is what's going to be mandated to the user back to the manufacturer. So what we say is the management station, the management station is shown on this box here shall get which is shown in the message being sent to the agent coordPatternStatus.0. So it's getting that one object. As we mentioned, the SNMP standard itself requires that the agent transmit the response once he receives a valid request. The reason we don't go into a lot of detail about that is because there are error conditions and other things that can happen. All of that is defined in SNMP. All of that is just kind of an automated response by the agent. We don't need to define that ourselves. So that's defined elsewhere within the

specifications. So setting the time and when you recognize, when we're defining some requirements and dialogs for some things if we've defined a requirement and we've imported that requirement from some other source we can probably import the dialog there as well. So the example of setting time we imported that from the CMS standard, the same thing. We can import the dialog from that CMS standard as well. It'll work just as well for that ASC. We actually took a lot of time within the NTCIP standards to make sure they were defined that way so that you could reuse code and that you would have reuse of patterns and things. So absolutely you should try to reference other standards when appropriate. And once again, we've provided a lot of this text for you within the student supplement. So we've shown you a couple of examples that are simple of setting one item. But what if it's more complex? Well, if you want to configure a timing pattern that's a whole bunch of information. The timing pattern not only includes the split between one phase and another but it also entails what's my offset from my coordination pulse? What's the cycle? And what's the split number, et cetera? So we'll go through those. When we look at it, we look at all of these objects. So this slide identified all of the different objects that relate to configuring a timing pattern. But we need to define them in a particular sequence and that's shown in this diagram. The first we need to do is make sure that we have enough splits, phases and patterns to meet our needs. But once we are confident that that is true we can start a process and for each phase we need a set of split numbers. We need to set the split time, what percentage of that time that particular phase is going to get for this pattern. We need to set the split mode which is it going to be resetting a vehicle recall or pedestrian recall when we're running those sorts of timing pattern and split. And then finally, the splitCoordPhase. What is your sync phase where for that cycling? And you'll notice here that each of these items have a dot-x, dot-y at the end of them. And that's because these are in a table. The first index in the table is the split number. The second is the phase number. So splitTime.1.1 would be the splitTime for split number one for phase number one. splitTime.1.2 would be split time for the split number 1 for phase number 2. So you just go on down the pattern that way. And if you have multiple splits supported by your device, which you probably do for timing patterns you'll have multiple split numbers and you'll have as many split phases as you have phases in your device. So you have a double index here. And you'll need to repeat the cycle for each phase for your given split if you're going to set your timing pattern for a particular timing pattern. You have one split and that split will have multiple phases. So for this one split you can use to repeat this process for each and every phase that you need to configure. Once you're done with all of those phases, then you move on to the next step of setting your pattern information, your cycle time, your offset time, your split number and your sequence number. And here you'll see that these are all ending with .z (dot-z). At the bottom that's defined as pattern number. So pattern cycle time for pattern number one is what I'm redefining. And you provide that information and that is your dialog. Now, once again, we need to convert that image into text so that we clearly define exactly what we're requiring to the manufacturer. So step one precondition, the ManagementStation shall confirm the consumer controller supports in the desired splits,

phases and patterns. That's simply what we define there at the top of the diagram. And then we translate each one of these messages into text. For each enabled phase, repeat step three. Step three is the ManagementStation shall set the following objects, desired values, splitTime.x.y, splitMode.x.y, splitCoordPhase.x.y and we define what (dot-x) .x and y are. Step four then the ManagementStation shall set the following objects desired values and we list out the objects we had in the figure and we define what dot-z is. So that's pretty straight forward. If we just keep on reusing the same basic structures in whatever combinations we need to in order to achieve our objective. So what are the implications of this? Well probably determined at this point that this is pretty detailed sort of information. But we also have to recognize that the procurement specification if we're going to be interoperable we need this level of specification. Otherwise there's going to be a high probability that the incompatibilities will arise because your central system may interpret it differently than the manufacturer did. So the dialog must be supported by both. And the dialog definition is a low level design issue and that requires some detailed expertise to define. So the other problem is that a lot of your existing projects that you're going to probably want to use off the shelf more or less, they've all ready implemented something. And you don't want to break necessarily what's all ready proven to work. You don't want to just come up with your own design, and then force them to change what they've all ready developed if it's all ready working. So really it's probably the management station developer is probably the best person who is qualified to develop this actual dialog for your system. They define how their system operates. And then you can tell all of the manufacturers of your devices this is how we want you to operate. Now, the catch here is that you may all ready have existing devices, which means the developer of your management system needs to make sure that your existing devices will support this dialog. Once again, that's probably a task best achieved by the developer themselves. This means that the developer is probably going to be chosen before finalizing your procurement for any new devices and care must be taken when upgrading your management station to avoid breaking your existing deployments. That brings us to our next quiz. Why should dialogs be defined in a procurement specification? And in this particular quiz, you can select multiple answers. So it's a multiple selection box. Any of these you should select, why should dialogs be defined in a procurement specification? Option A, devices are more likely to conform with the standard. Option B, devices are more likely to interoperate with the central system. Option C, devices are more likely to be interchangeable with other devices using the same procurement. Or Option D, devices are likely to be less expensive. So which one of those are true? Why should dialogs be defined in a procurement specification? Well, now that you've responded, option B is true likely to interoperate with central system. Dialogs promote a common expectation on how objects are to be exchanged and it promotes that interoperability. Likewise option C is true. It promotes interchangeability because once again it promotes that common expectation of how objects are to be exchanged. Option A was false. The dialog will be separate from the standard and will not affect the conformance. And option D is false. Additional specifications will likely mean added costs for the controller but the integration

costs should be significantly decreased. And that completes learning objective number two for achieving interoperability and interchangeability. We did review the terminology of what those terms mean. We discussed the various dialogs and objects and how you go about creating those dialogs. We prepared some sample dialogs and the text for those dialogs.

Ken Vaughn: Learning objective number three is to understand traceability and we'll talk about all of the user needs to requirements traceability and then the requirements to design traceability. And we'll also discuss some of the benefits of documenting this traceability. As you remember from the previous module, we started developing a user need traceability table. Of course, at that time all we had were the user needs. We didn't have the requirements. So we had a table like this with a placeholder there for C module A315b. Well, that's this module. So in this case, we're going to add a whole bunch of rows. We said we developed 20-some-odd requirements related to that one user need. So you can imagine we had all of these different user needs, underneath each one of those user needs we're going to have a long list of requirements. So this table becomes a rather large table. And, in fact, that's kind of the point we want to express is that these are very complex devices. You need to document all of the relationships otherwise something fails when you get it upon delivery and it's very difficult to understand all of the ramifications that sort of failure has. If you have a traceability table like this, you can quickly go back and understand the impact that will have on your project. That will help you to make proper management decision whether or not something can be deployed in the field or not. So this is really pretty critical to make sure you do this properly. And this is what it looks like. You just list all of the different requirements one after another underneath the particular user need that you're tracing to. Now, in comparison a very similar table exists within your PRL of standards that include SEP information. There are some distinct differences, though. You'll notice that this standard starts out, this table starts out with the user need, ID and user need name followed by requirement ID and requirement name. Those are also contained with the standards that have SEP information and what they call a per call requirements list. The distinction here is that we do not have a conformance or support column. Why? Because you're presumably developing this specifically for your project. Of course, everything for your project is going to be required for your project. So that kind of replaces the need for the conformance and support. Likewise additional and project requirements work for additional notes. In this context, it's easier to go ahead and write those notes into your actual requirement text rather than the separate area. There is no extra customization needed. You just put them straight into the text. Now, that's the first table to deal with. The second table is once you define user needs requirements, now you need to say requirements to my design elements, the dialogs and the individual devices or objects. So this table will be virtually identical to the table that is contained within SEP based standards requirements traceability matrix usually in Annex A of those standards. It traces requirements to a dialog as well to individual objects. And for each you typically give the requirement ID of

where it occurs within the text of your specification, the name or title of that requirement, the dialog reference, once again, clause reference to your specification for the dialog. And then for the object ID as well and then the object name. As an example, for a simple dialog you have a simple get coordination's pattern status. While the requirement was requirement 2.2.3.10 the requirement monitor current timing pattern. It's using the standard dialog, you'll notice that this is a reference to the NTCIP 1203. That's the CMS standard clause G.1 so Annex G sub clause 1. That's just a standard get dialog. We don't need to reinvent the wheel. We can go ahead and let them define exactly what the device needs to do there. And the specific object contained within that get request is the coordPatternStatus and we give the precise clause number as well. So that's how that one dialog would look in the PRL. Now for a more complex dialog you'll notice that there are a variety of objects listed here in different messages. The RTM does not specify where the objects are current messages. It's only intended to say if there is a relationship between the object and this requirement. If you want to see exactly the sequencing and everything then you have to look at the precise dialog. In this case, that's defined in clause 2.3.1 of our specification. And, once again, that's a reference to your student supplement where that's referenced. So you'd have your requirement ID, your requirement name, dialog references and then the specific reference for your object ID. And then, of course, most of your objects are going to be drawn in from different standards so all of these are going to typically be references to specific standard clauses and then the objects. So the benefits of traceability it is a fair amount of work to produce that information. But at the end of the day, you end up giving a purpose to every single design element. So once again, if you go and test your device and something fails you can very quickly trace back and identify what requirement has failed. I can look at that requirement, look at that text. Everyone can be happy that there's something missing. And then moving back on to look at the user need related to that requirement then I can understand that operational impact that little feature may have impacting my whole project. Also identifies very succinctly the objects that are required and those that are not. Once again you're implementing a standard but that standard includes lots of optional features. And by producing your own requirements traceability matrix you can quickly identify which objects you need to support in your devices and the ones that are not listed would not need to be supported. We also identify standardized dialogs. And so, once again, all of these allow you to very quickly when something fails in the field, when you're testing it, you can very quickly go back to this one table, identify where you are in that table and then immediately jump to the specific clause number of your specifications to define the exact dialog requirements, object requirements and everything else. We're dealing with hundreds of pages of specifications being able to quickly turn to exactly the right page is a real time saver. It allows you to solve and diagnose these problems in the field as opposed to having to go back and research them later and then you forget what the problem was. Everything is fresh in your mind this way. So you have the clauses of standards where details are defined, features impact if object is not supported, and whether an object is used in a test case that's required. And that last item is also very

important that we've seen before how some projects will use a test procedure that was actually written for a different project. And they deployed on this project because they look at the description of the test and they say well that seems to apply to my project as well so we're going to test it here. That test fails and they start investigating it and they realize, well, one object in here I'm returning no such name to and my device doesn't support it. And I look back through all of my documentation and I realize I don't even need to support that object for my particular project. That object relates to a different feature. I kind of understand why they put it into test procedure but it breaks my test. So actually it's not so much the device that's failing, it's the test procedure that's not applicable to my device in this particular instance. And so once again if you have that listing involved with your objects, if you've done all of your traceability it is very, very easy to identify these little anomalies and save your project a lot of time. So it is some work upfront to document everything. It is well worth it at the end of the day. The ASC specification will likely have hundreds of user needs, requirements, dialogs and objects. We don't minimize the effort we're talking about here. Instead, we're actually pointing out the fact that this is a major effort. We recognize some people try to do this sort of traceability in Excel spreadsheets and other things. The scope of this sort of effort really demands a more sophisticated tool. There are tools out there specifically designed for requirements management. We would certainly recommend using those sorts of tools on a project that is sort of large. I mean you're talking about hundreds or all together you're probably talking about thousands of items you're dealing with and it really requires a dedicated tool to manage that sort of environment. That brings us to the next quiz, which is not a benefit of traceability tables? They clearly identify the object associated with the requirement. Option B, they referenced the relevant clauses where items are defined. Option C, explains steps required in a test procedure. Or Option D, they provide traceability back to the user need. Once again, which of these are not a benefit of traceability tables? Go ahead and make your selection. Thanks a lot. All right, well the correct answer you see is option C. It does not explain steps required in the test procedure. Traceability tables do not address steps within a procedure. They only identify traceability between procedures and the requirements. All of the others are true. Option A, it does clearly identify the objects. They list each object required for each requirement. Option B, is true it provides references to relevant clauses. Each item is associated with a reference to the clause where the item is defined. And finally, Option D is true, a user can trace from object to user need in either direction. That concludes learning objective number three understanding traceability. We did talk about the user needs requirements traceability table. And it's a simplified version of the protocol requirements list. We also talked about the requirements traceability matrix which is identical to that in the standards that have SEP content. And we also talked about the traceability takes effort but it is well worth it.

Ken Vaughn: And that brings us to your learning objective number four, the last learning objective for this module portion. And we're going to talk about how to develop the specification and how you put all of this text together. We'll provide checklist of key

elements and we'll also talk about how the NTCIP specification fits into the overall package. We'll address that last one first. Developing specification, making sure that your interface specification is only one portion of that larger package. And that your interface specification needs to be consistent with your hardware specification and your software specification. It's one part of that specification and it's just part of a larger package. It is also likely to that you will have more than one interface specification. If you're defining your procurement for a device then you're very possibly going to have to deal with not only where you are today but where you want to be in the future. So you may have a proprietary or a legacy sort of protocol that you're defining as well as your new NTCIP protocol. And, of course, if you're defining specifications for a central system even more so. You will have almost certainly multiple devices and you'll have a different interface specification for each different device type, maybe even for different versions of that device type. And also perhaps some legacy or proprietary protocols there as well. Each of those interfaces need to be defined separately and they're recorded separately. So but they'll all be required by your device or your central system. And, of course, what we've talked about is only one portion of your interface specification. It deals with the data dictionary section and your dialogs and such but you also need to require the communications stack and that's the subject of the C.1.1 module that we discussed earlier in the curriculum path. Moving on we'll talk about the interface specification some. There is sample text. Once again, we strongly encourage you to go ahead and download the student supplement from the available resources. And the example text in your student supplement includes a boiler plate. It includes sample user needs, sample requirements, sample dialogs. A lot of this stuff we pulled in from other standards. We've also provided the samples that we've used in this presentation, all of them in the student supplement. It also discusses custom objects, user needs to the requirements traceability table and requirements matrix in the communication stack specifications. So that's everything that goes into that sort of interface specification. And then finally that brings us to our last activity of the day. Which of the following statements are true and once again this is going to be a multiple answer capability. So if there are multiple true answers, go ahead and select both of them or three of them or four of them. Option A, the interface specification is the most important part of a procurement. Option B, an interface specification should contain or reference dialogs. Option C, an interface specification must define the communication stack. And option D, a central system should only support one interface. So go ahead and make your selections. Once, again, multiple choice option this time and then we'll talk about that here in a second. Well, hopefully you got this right. Option B is correct in order to achieve interoperability the dialog should be explicit. And option C is also true the communications stack defines over what medium components it will communicate. Option A is incorrect. The schedule, budget, hardware and software specifications are all important. And option D is also false. Most central systems will need to support at least one interface per device type. That concludes learning objective number four developing the specification. We talked about the interface specification being part of a larger package. And we also talked about the fact that the component may

need to support multiple interfaces. And there were also several parts to an interface specification. Well, today we've learned that requirements can be identified from user needs, SEP based standards and conformance groups as well as the three different perspectives of monitor, control, configure. Well written requirements clearly identify the actor, action and target of the action. And dialogs define roles on how to exchange your information while objects the define the meaning of information. Both must be defined to achieve interoperability. Traceability tables allow a user to quickly determine why a design feature is included. And finally, in the interface specification should reference standards whenever possible rather than copying their content. Well, that concludes the body of our presentation. There are other resources to be aware of. There's the NTCIP 1202 standard itself that was published in 2005 and that's the Object Definition for Actuated for Traffic Center Controller version 2. There's also the NTCIP guide that is NTCIP 9001 the most recent update was version 4 published in 2009. Both of those are available at NTCIP.org. There's also the IEEE standard related to requirements engineering that is IEEE 29148 published in 2011 and that's available via ISO. Now, that comes to questions. We have heard a few questions from this presentation and that includes are there any 1202 procurement specifications? Well, there have been some projects, nothing to this sort of level. We do need some projects to go forward and actually do this sort of really high quality specifications. You can get some input, though. New York City did a major procurement and they would probably have a lot of information that you could learn from. The Federal Highway Administration, of course, is always a good resource. You can go contact the resource centers, et cetera. And, of course, as we've mentioned before the participant student supplement that is available on the resources area of the same location where you downloaded this. Another question we've heard is how we can get assistance. Once again, resource centers are an excellent, excellent resource. You can contact the operations team of the resource centers through the Federal Highway Administration. Also ITS peer to peer program is another resource where you can find other agencies out there that have faced similar issues and get their input. Finally, we've also heard that people are wondering how they can get additional training materials and that is available from the same site where you downloaded this information www.pcb.its.dot.gov. So those are the main questions that we've seen today and you're welcome to contact us otherwise at the resource centers to provide more information. And I hope you enjoyed this and with that and we close off the seminar and thanks a lot. Bye. And there is, in fact, one additional course module of this, A315b Understanding Requirements for Actuated Traffic Center Controllers Based on NTCIP 1202 Standard part 2 of 2. And in that course we will go ahead and discuss the other two learning objectives that we mentioned at the start of this course. And then beyond that, of course, there's the testing module as well that we discussed earlier in this course. With that, thank you for attending.

End of A315b_final.mp4####