T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

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 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 at www.pcb.its.dot.gov. Please help us make even more improvements to our training modules through the evaluation process. We will look forward to hearing your comments, and thank you again for participating, and we hope you find this module helpful.

Narrator: Throughout the presentation, this activity slide will appear, indicating there is a multiple choice pop quiz following this slide. You will use your computer mouse to select your answer. There is only one correct answer, and selecting the Submit button will record your answer and the Clear button will remove your answer if you wish to select another answer. You will receive instant feedback on your answer choice. This module is T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan. Your instructor, Dave Miller, has more than 36 years of experience in the management, design, and development of the critical control systems, instead of traffic control, bus rapid transit, and connected vehicles. Specific ITS experience includes the design and the first deployments of ATC 5201 and ATC 5202. ASCs as experience includes the design and first deployments of NEMA TS-2 ASCs with NTCIP. Connected vehicle experience includes the ASCs and the software for the USDOT test bed in Oakland County, Michigan, and the U.S. Safety Pilot in Ann Arbor, Michigan, and test beds in the U.K. and the European Union.

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

Dave Miller: In this course, our target audience is shown on the slide here, and this is targeted towards test personnel responsible for developing test procedures, maintenance staff that use test procedures to troubleshoot the equipment once it's out there. Audience here is also system developers, procurement people that are writing contracts, and private and public sector users, including manufacturers, would be interested in this course.

These are the prerequisites. We're not going to go through them, but as you recall, if you got to this course, you've probably been with the T series, starting with T101, that began with an introduction and T204 Part 1 of this course, and again this is a second part of one course called T204: How to Develop a Test Procedure for ITS Standards.

Again, we're not going to go through in detail, but this is the curriculum path. So this is what you've gone through, and then in the lower left there is where we are right now. So this is Part 2 of two of this course. We're going to be using quite a few acronyms here, so we're going to list these upfront. Again, we won't go through these in detail. A lot of these you're familiar with center-to-field, dynamic message signs, management information base, et cetera and when we go through and we hit one, we'll have it in parentheses so we can reference it as we go along.

And again, in the prerequisite testing modules, we were given an overview of the testing workflow, and it was followed by two modules dedicated to developing test cases. So in Part 1 of this module we moved on to developing test procedures and began with the purpose and structure of a typical test procedure, and then we learned the role of documenting a test procedure using a written specification, the importance of synchronizing the test specification to the contract terms so that everything comes out correctly at the end. Finally, we learned to write reports, and those are produced at the end of testing. We did a sample TSP structure and showed how to use standard tools.

At the very end of Part 1, the audience was encouraged to obtain a copy of the TPG, the Test Procedure Generator, which will be used extensively in Part 2 here. So as you recall from then, it was a computer tool. We'll go through it again, and we gave you an opportunity in the break between Part 1 and Part 2 where you could download it and play with it a little bit. We will use the TPG to develop test procedures for a variety of different equipment for the roadside, center-to-field. A second learning objective of this part involves adapting those generated test procedures to the contract terms and conditions; and in the final learning objective of this second half, we will develop complex text procedures that will pull together NTCIP elements using the TPG. And remember, as we

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

talked about before, the primary object of the TPG is to provide consistent methods and tools to assist the user in development of test procedures that verify conformance with the NTCIP center-to-field device interface standards. Second objectives are to reduce development risk, effort and cost of developing test procedures, which in turn will encourage the development of interoperable NTCIP systems.

The Test Procedure Generator, or TPG, allows a user to develop test procedures from the NTCIP center-to-field device interface standards that have a Management Information Base, or MIB, and that exchanges dialogs from the center to the field and has a Requirements Traceability Matrix, called RTM, that defines conformance. So we'll find here as we go through this that agencies can use this TPG to generate test procedures to help determine an implementation's conformance to the NTCIP Interface Standards or to the compliance of a project specification if extensions to the standard have to be added for the project. So that's kind of where we're headed here in Part 2.

We'll start out here. We're going to a slide from Part 1, and we'll go through this again. This is what we learned in Part 1. So we learned that the test procedure, TSP, inputs our test cases. So the inputs to the specification are test cases that are used to trade outputs of expected results and anomaly reports of unexpected results in IEEE 829 format. We also learned that a test design specification details what a test is to demonstrate, while a test case specification specifies inputs, the predicted result, and conditions, while the test procedure defines the actual steps that are needed to perform the test.

In Part 1 we also learned that contract terms and conditions should be viewed from the project's end, including test case specifications and test procedures. If you develop those upfront, you can include them in the terms for deliverables. A Master Test Report measures project success to the stated goals, and finally, we learned that, in Part 1, the Test Procedure Generator is an automated tool that generates XML scripts using consistent keywords for interoperability.

So that's kind of a very quick review of Part 1, and we're going to just review quickly this diagram of the workflow just to show you where we are in the test workflow. So this diagram on the right-hand of this slide is the test workflow, and the initial test plan at the top. If you recall in our previous modules, we developed the test plan. The test plan is followed by test design specification: the details, the approach in the test, and then if you follow down, the workflow is then broken down into individual test cases. It has inputs, expected result, and the execution conditions. So the test procedures document step-by-step procedures. So down here at the bottom, on 204, the procedures down here

document the step-by-step procedures. So these were the cases—we had a specification of cases—but this is a step-by-step procedures down here that the person running the test would use. And again, recall that we said that an individual test procedure may cover more than one test case, so we might have—our simple example from the first part of this module was say we were testing yellow operation; four-second yellow operation was the test case and another test case that tests red, for example. You might write a procedure that would take the same equipment and write a procedure that would test both of those in one procedure. So the cases are individual, but you can write a procedure to swallow up several cases at once. So this completes our brief review of Part 1 of this module.

So at this point, we're going to begin Part 2, which is to complete the remaining learning objectives that we laid out in the beginning of the module. So we're going to continue on now to the Learning Objective 6, and this is where we will begin to use the TPG to generate test procedures for a variety of equipment. So what we're going to start doing here is begin creating simple examples for actuated signal control, dynamic message signs, central station, and others. And for each of these equipment types, we'll reference the appropriate NTCIP standard. And you recall that we recommended in previous modules that you begin with the bottom-up approach for center-to-field communications. NTCIP standards can appear to be overwhelming, but really you go to the end of them, at the bottom, and you can see the different actual objects and you can see conformance groups in a lot of them. So if you start at the bottom with the procedures, from the Table of Contents, you'll always start out with something that's familiar and you can work up from there. For example, we're going to kind of go through a dynamic message sign, DMS sign example here, using the TPG. So we're going to start at the bottom with the conformance group within NTCIP 1203, which is the DMS standard, and then we're going to enter some examples into the TPG for what the dialogs will look like.

We're going to talk on this slide about NTCIP 8002, which is the document that should be used by all NTCIP documents to define the content outline, clause numbering, including section numbering, clause and sub clause numbering, subject content, annex numbering and labels, tags, and subclasses. That sounds like a lot, but as we know, the NTCIP standards are complex and it's good if they would all have the same format. So if you looked at any standard for any of these equipment at the roadside. You pick it up, you look at it, you say, "Oh, the table of contents looks right. It all looks familiar. The clause numbering is consistent." So if you're going from one standard to the other, they have a consistent format. So this is kind of a standard of standards. This is a standard that a person that was doing the final editing on any of the one of the standards would say, "This is the format you would follow so it always looks consistent among them." So in short, 8002 imposes a standard numbering system and content format for all the NTCIP

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

1200 series standards, like 1202 for actuated signal control. And if you go on ntcip.org and you look, you can do a search for NTCIP 8002, you will find that at this time right now, when we're presenting this, that it's really not published and available for download but it is used by the working groups that are developing the standards. So it's not really out there for distribution, but if you see a reference to 8002 and you're on a standards committee, that would be what you would follow when you're sitting at the word processing and you're putting in all the words and the formats so they're all consistent. The reason we're bringing it up here briefly is that the TPG verifies that the standards used in each session are conformant to NTCIP 8002 numbering system and content format. So you'll see as we go through this, your test procedures, if they're created by this TPG tool, it conforms to the standards. That's a huge help, and TPG references the standard and as you'll see as we start generating these procedures, you will see that it automatically generates them to match the standards so that they're familiar for everyone that uses them.

This is the role of the TPG. So what is the role, in terms of the overall testing process workflow that we've learned up until now? The TPG guides the development of test procedures for selected requirements of NTCIP center-to-field standards. So let's just back off a little bit and say if you're a standards developer, you can import the proposed draft standards. So we have working groups; they develop the standards and there's a draft of it. You can import the proposed draft standard into the TPG and it will report noncompliances to the format that's prescribed by 8002 Annex B1. If you're not a standards developer, and I assume most of you are not, TPG will ensure that the standard you are using in your project has a common understanding of the keywords, what the objects mean, and traceability back to requirements. That is very powerful. If you're trying to write a communications dialog, and putting the payload in, and trying to match the requirements, it's absolutely essential that both ends of the communication links interprets the wording the same, the content, and all of that that goes with the protocol identically at each end or it won't work correctly. So the TPG is a very powerful tool. Its output can be used in the beginning of the project so that all stakeholders can test throughout the project to avoid surprises during system integration level. So it's really good that everyone that's working on the project, developing the objects and the dialogs from the central—like the central supplier, the roadside equipment supplier—if they all use the TPG, they can have that tool and as they're developing their dialogs to meet the requirements, it will automatically make sure that they are interpreted the same way and are labeled the same way.

I put a little table down here. This table—I know a lot of you have probably downloaded this, and this is included in the manual, but I just put it there to show what this is. The

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

TPG requires a software interface. So the latest version of TPG that's available for distribution we'll give you the link for distribution again later on in the slides. The TPG requires Microsoft Windows 7 operating system with at least 2009 service pack 1. It also requires MS Word 2010 Professional Edition, and if you look in the manual it'll have additional information on the hardware platform, like memory-pointing device, whatever. But it's a basic typical Windows platform. So if you have this, it should install correctly and run correctly.

So again, between Part 1 and Part 2 of this module, a lot of you have probably finished installing the TPG using the install shield wizard, and you should have a TPG shortcut placed on your Windows desktop as shown here. So this is what it looks like. You'll be able to see an icon on your desktop to click it to launch the TPG if you ran install wizard and got everything installed correctly on your laptop or whatever platform you're using. What it does is it'll put the shortcut on your desktop and it'll also create an empty directory named TPG Standards in your primary volume, which would be typically your C drive. So once you've installed this and gone through the install wizard, you should see the icon on the desktop. Go to your C drive and look and see what directories, you'll find one called TPG and it will be empty. So if it was installed correctly, that's the point that you should have gotten to if you're going to follow along here.

So if you have it up and running, if you go ahead and launch it, the GUI or graphical user interface looks like this, and it's pretty familiar to anyone that's used Windows GUI. It includes the TPG module across the top here for all the menu items, across the top banner, and the document tabs are just below it. So these are the banners, the menu items, and then these are document tabs. So these are all the documents we'll be working with.

The main window down here, the main white window here, shows the NTCIP standard as an embedded MS Word 2010 document with the MS Word document menu just above. So that's what's nice about it. If you have the TPG, and say we're working on NTCIP 1203, we're working on a dynamic message sign, you can pull that up. It's going to find the correct, most latest adopted version of it, which is very nice. In this case, it brings it up as version V03. The session panel appears as the left-most column while the TPG Status and Command is here across the bottom. So this is what they call a session panel, but this is all the things we will be doing, the workflow, and the session status is down here. So this is what's going on at the moment. So that's a brief run through the GUI, and again, if you load it and launch it, it'll all become very familiar quickly. So go through the manual with the description of what the different parts of the screen are.

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

So here we're going to start an example. So we're going to begin by resetting the TPG environment to clear any previous steps that may be left over from previous sessions. So the items reset are shown. So what we did is we went to File, Options, Reset Default TPG Settings. So when we do that, if we were in the middle of another session or something was left over, this is going to reset the defaults, put us back to be able to start a clean design here.

The next thing we're doing is we will begin a new session by going again to File, New Session (like is shown here) and then what it's going to do, it'll go ahead and open a new session using the standard that we picked in the previous slide. The response to that is now you get a screen that looks like this, and we select—in this case, we're going to select the center-to-field standard number that matches our equipment. So it's going to come up and say, "Okay, here are some standards." We're selecting for, this example, NTCIP 1203, which is shown here 1203. The 12 is always fixed because TPG works for the 1200 series, 12xx series standards. We're going to put an "03" here because we are going to do a dynamic message sign, which is 1203, and then it shows major and minor revisions. So when we're going along here at this point in time, NTCIP 1203, it's version 3.01b. So that's how it gets entered in here.

This is probably the most common way to open a new session. If you read the manual, you'll read there's other ways to open the session. This is probably the most common way. You reset it, you start a new session, and you fill it in it here in this fill-in screen. You can also open the most recent set of test procedure from a new session's window. Again, if you look at the manual, you can open up an old one and a new one, and there's different ways to launch it, but this would be the most common way of starting a new one.

Moving on, after pressing Okay on the previous screen, the TPG will first read in the standard document and then read the table of contents. So once you press Okay on the previous screen, it's going to say, "Okay, I'm going into that document, NTCIP 1202 version 3.01, and I'm going to start reading that document." So it's an MS Word document. It's going to start reading it in. It's going automatically into the MS Word document and it's pulling in the table of contents. So this table of contents will automatically be compared to the document to make sure the table of the contents and the document match each other. So test cases are loaded later. I mean, we're going to show how to load test cases later if the standard's been opened. What we're doing here is we're making sure we're starting with the correct latest adopted version of the standard. It's going to make sure we're all using the same standard. If I am at the central and somebody else is at the dynamic message sign; we use the TPG at both ends but

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

the same tool so we're always starting with the same document and the same table of contents. So from the prior screen, it was Command in Progress. So it was in progress in the prior screen, uploading the table of contents from the MS Word document for that standard. Once that Command in Progress window disappears, it comes up with MAP NTCIP 8002 Required Sections window, as shown. And recall from a couple of slides ago, NTCIP 8002 provides the standard numbering system and content for all of the 1200 series standards. If the NTCIP 1200 series standard contains sections defined by 8002, they're automatically mapped to the section on the left of the screen as shown. So on the left side, it's going through and it's finding—going through the document that we selected, the table of contents. It's checking it to make sure that it matches, and if it matches, they're automatically mapped on the left side of the screen. If the 1200 standard does not contain sections that conform to 8002, you'll have to map them by clicking on the Click Table of Contents entry to set new section button. And this is kind of a little extra work, but NTCIP has been around for a while.

The newest stuff and the ones that are going through five-year reviews are brought up to 8002, so little by little the TPG tool is eventually going to automatically select them all as the working groups bring them all up to date to the standards. If you run across one that isn't, you might have to pull up the table of contents. You may have to map it manually by using the mouse and clicking it. If that Click Table of Contents is not active, the sections all map correctly and you don't have to do anything manually. So ideally if you have a standard that the working groups finished and it's up to speed, so it conforms to 8002, it's going to come across. You won't be able to actually do anything else. It'll automatically bring things across. If it finds something you can't bring across, it'll un-gray that button and you can map the rest of the sections manually.

So the left side of the screen shows Annex B1 required sections and the right is what the loaded standards contain. So you can see that these are the sections from our center-to-field standard, concept of operations dialogs, on and on and on, for the standard selected. So in this example, you can see that the TPG found section 4.2 and it did a lot of the mapping. So you can see that the message states that it conformed to 8002, was verified and was unable to verify one or more sections. So we'll have to click on the table of contents entry to do a few more sections.

So moving on here, here's where we see the results of the automatic mapping. So it automatically mapped everything it could, and for a dynamic message sign example, we had to map a few things by hand. But once we get through that, the mapping is complete. So we can see ıf the TPG is identifying issues that need to be dealt with in the standards

working group. So if I was a person in a working group—say it was one of the standards—let's say it's a five-year review period and we're reviewing it—we would run that standard through the TPG and it would automatically map some of it. It would find some things that don't conform to the standard numbering system and the way things are laid out, and the working group would take care of fixing that so that eventually they all come in automatically. So that's another powerful example of the TPG for the people that are writing the standard. It's automatically going to force them to use the same format and understanding. So again, over time all the working groups will be in 8002. You'll still run into some you'll have to map by hand, but you should be able to at least bring them up and you'll bring in the sections on the right side of the screen and you can do the mapping.

The TPG verifies the content of the mapping and then it moves on and proceeds to construct the Requirements Traceability Matrix with the display of the percentage completed. So you can see here that it says 4 percent, but during this process the TPG is now reading in all of the requirements, the objects and dialogs found in the standards and in the Requirement Traceability Matrix. So any errors in the standard in RTM are detected as the document is read. So at this point, while it's reading it in, the user is starting with the Requirements Traceability Matrix that is based on a complete and correct standard. So that's very important that we're all starting with our requirements, because we want our user needs traced to requirements, requirements traced to design. We've been through that in prior modules, but just make sure that the Requirements Traceability Matrix is based on a complete and correct standard so that it works for every piece of equipment that's on the communication link. And you might say, "Why are we going through all of this?" Well, you'll see as we go along, this Requirements Traceability Matrix is used as the basis for our testing. So this is Part 2 of testing, so we're testing to requirements. So what we're doing is starting with a correct standard, we're bringing in the Requirements Traceability Matrix and we're going to use that as our basis to start doing our test procedures. That way, all parts of a design are tested on an identical standard and on identical common understanding of each standard by each project stakeholder. So if I am the agency that's going to procure this, I have the same understanding as the person that's designing it, from either the manufacturer or the person that's going to install it and test it, if you're a maintenance person or you're going to maintain it over the years. So a powerful part of the TPG is the project will then be successfully delivered and deployed with minimal open issues. Again, we're trying to get to the end with no surprises. So this is a very powerful tool to make sure that we're beginning with a standard and we're linking all of our requirements to it, so at the end when the design is done and we put it on the street there won't be surprises.

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

We're going to go ahead and do an activity here. And the activity here is: How does the TPG relate to the NTCIP 8002? So we have some answer choices here. So let's go ahead and answer what you think the correct answer here is. So “A” is the draft standards are verified to NTCIP 8002 compliance. Choice “B” would be it relates as an unbroken traceability from requirements through testing. Answer “C” is it relates to create uniformity of test procedure content and numbering. Or the choice “D” would be all of the above. So we'll go ahead and open up the poll and you can go ahead and select your correct answer. We'll go ahead and review the answers now. The correct answer is “D:” all of the above answers describe how the TPG relates to NTCIP 8002. “A,” draft standards are verified to 8002 conformance. We saw that when we ran the TPG and we brought the standard in and it said, "Everything's conformant. The table of contents and everything is correct to the standard," or it would find some that weren't and you would have to map them manually. TPG provides unbroken traceability from requirements through testing. Again, we saw that, that we had a Requirements Traceability Matrix that we picked up and it was going to carry that forward through testing. And “C,” uniformity of test procedure content and numbering. Again, we talked about how the TPG provides uniformity so that it matches the standard way that the contents and the numbers are created so that all the standards are uniform across the 1200 series standards.

Following that exercise, we're going to move on to an example for a dynamic message sign. So from our previous screen, the Requirements Traceability Matrix process reaches 100 percent. Recall before we did this poll the last screen was creating a Requirements Traceability Matrix. So we're assuming here that it reached 100 percent, the window was closed and the appropriate standards display in the Standards tab that matches the standards selected when the session was opened earlier. And again, as you recall, we're using a dynamic message sign example, so we selected NTCIP 1203 version 03b, as you can see there in the box over here.

So now we're going to start creating a new set of test procedures. So to do that, we select File, New Set of Test Procedures in this command, as shown here, in the pull-down menu, and once you click on that it will come up with this window that says I am creating—begins creating a new set of test procedure file and the TPRTM file, Test Procedure Requirements Traceability Matrix file. And we learned in the prior modules that that file is built by the TPG and it creates a table of all the requirements found in the Requirements Traceability Matrix for the selected standard and it maps those to the test procedures we want to create. So here it's going into dynamic message sign and it's going to keep running until it brings in a complete traceability matrix so that we lock our requirements to the test procedures we're going to create here. After the test procedures is created and saved, the test procedure name and ID are added to the table

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

corresponding to the requirements rows that the procedure tests do. So you can see that it's going to go ahead and pull all these in.

This Set of Test Procedures file is displayed at completion. So the Set of Test Procedures file is displayed on the tab and the standard remains open in the Standards tab. So here you can see that we've gone back. We're still looking at the front cover of our standard here and the set of test procedures has been pulled in, so we're going to move on to the next slide here.

So now that the TPG has a set of test procedures in the traceability matrix, we can now begin to create each test procedure for our dynamic message sign design. So what we're going to do here is shown. Go into Test Procedure, New Test Procedure. So at this point, the TPG in the prior slide has found the correct standard, brought in the latest version of it, pulled in the table of contents, looked at the table of contents and decided what was conformant to 8002 standard automatically. Any that weren't, you would map them. You would go into it and map them with the point-click of your mouse. We pulled in our Requirements Traceability Matrix and we're ready to start writing a procedure to test each requirement at this point.

So an example test procedure. So from the prior mouse click on the prior screen, the TPG responds with a test procedure identification number, so that's the window here, and you can see this test procedure ID. It comes up in this little window. It says Test Procedure ID. This is going to be Test Procedure number 1, 01.00. So it's going to automatically assign those numbers so they're all unique and different from each other. So if you remember from our prior training modules, we learned that test procedures can be used to verify one or more test cases and inputs and that each test procedure must have a unique ID number and is populated by this screen. So again, it's just going into Test Procedure Number to test a portion of the requirements. So when we looked at the last one, we said, "Okay, we're starting to write test procedures. That's test procedure number one. That makes sense." So we said Okay there. The TPG responds with this screen, which is the actual test procedure document, and it's stored to a file and shown here. So when we said that, we said, "Okay, the first test procedure is number one in the previous screen." We clicked Okay. So it's actually going to come up automatically and show you this nice format for your first test procedure. So you can see it's populated with test procedure identification of 01.00, and you will see that it automatically adopts the format in this test procedure of Annex B1, meaning that users may not directly edit the document from this screen. The test procedure must be edited using the TPG menu's functionality. So it's going to come up with this test procedure and it's going to show you

this format and you're not going to be able to change it because it's going to say, if we're following all the standards, "This is what a test procedure should look like." This is the format of it, and everyone that's using the TPG will see the same formats and be familiar with them without having to ask any questions on what is the meaning of things.

So if we click in this screen, we're going to click Define Test Procedure Header. So you can see here—so we're going to actually go ahead and define what the header says, so that's what we did over here in this pull-down menu. So the TPG, after we clicked on the prior menu with our mouse, the TPG responds with this screen and this has some entry of locations that we need to populate. The title of the test procedure, the description of the test procedure, and the test procedure pass/fail criteria that is going to be used by the test operator. So this comes up in this case it's dynamic message sign test procedure number one—we populated already—and we have to fill these in so that the test operator has that information. And remember, pass in a test procedure indicates the device under test—in this case a dynamic message sign—produced expected results when you ran the test, while unexpected results require further investigation. If something comes back in your test that wasn't expected, that's not necessarily a fail; it just means we have to investigate further. Unexpected results could be a defect in the design, or it might even be a defect in the test procedure itself or a defect—or a failure of the test operator—the operator has tested it wrong. It happens. So if the results are expected, pass. If the results are unexpected, more work, more investigation into what was actually causing it. We never jump to the conclusion that the device under test was a failure without verifying that. So the value of the TPG again be seen here, that a failure of the test procedure is eliminated. So if we were doing this all by hand without the TPG as a tool and we were doing a test procedure and got unexpected results, we would have to go back and look at how we wrote the procedure and did it conform to the standards, on and on and on. It can waste a lot of time. But this one, you can almost immediately say, "I read it on the TPG. It selected everything correctly." So the TPGs rule out that the unexpected result was caused by the procedure itself.

So an incorrect test procedure can still be created, but the TPG assists to make sure the keywords are correct, formatting is correct, the syntax is correct, but the user will still have to structure each of the steps. So this is making sure that this procedure number one is correctly done, but if we strung several test procedures in the wrong—or we did the procedures in wrong steps or something, we could get unexpected results, but we could still—came from these procedures that were done by the TPG, and we'll show that a little bit here as we go along.

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

From the prior screen, we populated the header information. So you can see our very first test procedure appears as shown here. So you can see this is Test Procedure 1. The title of the procedure is Determine Sign Type and Technology, and it's got a description: Test case verifies DMS indicates the design type and uses the technology required by the specification. So again, this comes up with the fill-in-the-blank boxes and it puts them in the proper place, and you can rest assured that what you see here on the screen is in the proper format and it'll be familiar to everyone that's going to be using it later on and forever.

Next thing, we will select the requirements for our dynamic message sign device. So that goes Test Procedure pull-down menu and select Requirements. TPG responds with the list of standard requirements with the checkboxes used to select the requirements of our message sign device that we're doing here. The requirements shown here are all loaded from the NTCIP 1203 Requirements Traceability Matrix. So this standard, 1203, has Requirements Traceability Matrix and it's all figured out already. So it's going to come up and show you all of those. So this goes into the standard and these are possible requirements.

So again, in this message sign, the example, we have selected checkboxes. In our design we're going to retrieve data, so that's over here. We're selecting, as you can see, that we'll check some of these. We are going to have requirements to retrieve data, requirements to deliver data, requirements to explore data, and determine current configuration of logging service, configure logging service. So note that once those are checked, the TPG retrieves the correct section number of each requirement from the standard. So if we check those, it's going to have the correct section number, so we don't have to go and delve into the standard and see what the section numbers are and fill them in manually.

After selecting the requirements, we can see here that the test procedure is populated with the selected requirements as shown. And again, the TPG populates the procedure with the correct section numbers per the 8002 standard. So you can see that it's actually put—this is a test procedure to verify these five requirements: retrieve, deliver, explore, determine, and configure. Again, it's nice that the TPG brings these in with the correct section numbers, spelled correctly, puts them in the right place in the right format. So everybody that's familiar with using the TPG will know where they are and what that means. It's all very understandable.

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

Next, we are going to populate the test procedure with the variables of our DMS design. So here you can see that we used the pull-down tab to select the Define Variables tab. So again, it's Test Procedures, Define Variables, click on that. And once we clicked on that, the TPG responds to the screen with fields to enter the variable names. So this comes up and says Enter Variable Name in this screen, and it's going to respond with a window with fields to enter the variables as shown, which are, in this case, our variables in our design that we're going to—that are going to be unique to our designer-required sign technology and required sign type. So we're using those two as examples here. And the user has the option to select other objects for variables. But just for a simple demonstration, we're going to select these two.

So you can see that after we did that and moved on, we can see that the variables are populated, and our DMS test procedure document is shown here. So now we have ID number, name of the procedure, description of the procedure, the requirements that we're testing in our design, and the variables that we're going to use as inputs. These are the things we're going to change. And again, they came from the standard so they're spelled correctly, they're shown correctly, and there's a common understanding, and it eliminates lots of possibility for error if you had to enter all that manually into a test procedure, like in a Word document.

Remember the test procedures include the testing workflow of the sequence of test steps. So the procedures are really the sequence of tests. The operator says, "Do this as Step 1. Do this as Step 2. Do this as Step 3." So the TPG now displays a window to populate the test step number along a standard keyword, and in this case you can see there that—it says Step 1 of a "get" of a dynamic message sign data. So we can say our keyword here is "get," and that's a standard keyword and it's used to get data from the DMS.

For our selected variables, the TPG presents a list of valid objects based on the requirements for our design. So you can see here the list on the left, over here, contains the objects that can be used based on our selected variable allows us to create the options we want to select over to the list on the right. So these are the ones that are available, and you can drag them over to the right if you want to use them, or as you can see up here, you can use your mouse to drag and drop the order on them. So you're just trying to put the ones over here that we want to use.

After dragging the objects on the prior screen and we did okay, the TPG returns to the previous screen where can click the Insert button where the data is added to the test

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

procedure as shown. So we did an Insert, and you can see up here on the top we had the variables. We did that prior, but after we've done that, this is actually going to automatically populate the test step. So Test Step Number One, this is the procedure. It puts it all in there nicely and it's all done correctly and spelled correctly from the standard.

After the test procedure is fully populated with our design data from the variables, we can save the test procedure using the pull-down tab as shown. So you can see here we're just saying Test Procedure pull-down tab, Save Procedure. So this shows the procedure. So we've got a procedure with one test step in it. Make sure you're aware that the TPG has many other editing functions. I mean, we're not going to go through them all here for time, but if you've downloaded and installed it in the install wizard and you have the banner, you can see that we've just created a procedure and we've populated the procedure, but there's lots of other things you can do. You can—lots of other editing functions. Like you can do Delete. You can re-sort them to a different order. You can renumber them in different orders. And again, refer to the TPG manual for more information.

When the test procedure is saved, it's added to the set of test procedures. So this is one test procedure, and when we save it, it adds it to a set for this design—set of procedures that we're going to use to verify this message sign. At this point, once we've saved it, we can go ahead and close it. So that's pretty basic. Close Test Procedure, and you do the pull-down tab and it'll close it. So at this point it's been—our work has been saved and it's archived away.

Again, recall that systems typically consist of multiple test cases. We saw that in one of our early slides here. We had our diagram of the test workflow. There's usually multiple test cases and multiple level tests that are part of a master test specification. We learned that in the prior part of this—Part 1 of this. TPG allows these related test procedures as sets, again using the pull-down tab. So we just created a test procedure, we saved it, but as we found in the beginning, if you're going to test a complete system or in this case a message sign, you're going to have a number of different test procedures and those are saved as a set so that you can bring them all up in a group.

So when we save this in the set, the TPG responds with a Save As screen that allows us to name and save our related test procedures as a set. So it's going to say if we want to put these in the set. This all leads into a file, filename, save as type. The type is set of test procedure. So if we had several test procedures, it'll save them all as one set. So if the operator comes later and says, "Oh, I got to test this thing again," just bring up

everything. "Bring up the set of test procedures I need to run through for running the test again."

Recall that our test sequence requires test documentation as outputs. We talked about that in Part 1 of this two-part module. The goal here of the workflow is to have test documentation as output of these procedures. So the TPG creates the output documents by saving the test results as XML files. I think we talked about that in the first one briefly. When we're running the test, we get expected/unexpected results, whatever, when we run the test, the output is saved as XML files, and to perform this function, the standard and the set of test procedure file must be open. So what we want to do is save the output as an XML set of test procedures. So we have to have it open, we have to have NTCIP 1203 because we are doing a dynamic message sign here, and the XML files are created by selecting File and Save. So it's File, Save, XML set of test procedure. But again, make sure you have the correct one open when you do that. So the XML representation can be displayed using the same drop-down menu. So you can see that we saved it. You can also display it up here.

So when you do that, TPG automatically switches to the Reports tab to display the XML representation. And again, as we learned in Part 1 of this module, this XML representation can be used as input scripts to automatically test equipment. So this is the result here. Very quickly, we found the standard we wanted to use, the TPG pulled the standard in, checked the table of contents, found anything that wasn't conformant to the standard for the table. We dragged and dropped the ones that weren't. We did an identification, we got to name it, we put in the Requirements Traceability Matrix, we had the step-by-step procedure, we saved it as a set, and then the end of that is this is a script that can be used to actually run the test automatically, and this is in standard XML format.

We're going to do another activity here. This one we're asking the question, "What is the order of the TPG example workflow?" So as you can see, we have, again, a choice of four answers. Answer “A” is XML—requirements follow the XML—requirements, step, object, variable. Is that the actual workflow? Do you do XML first, then the requirements, then the steps, then objects, and variables? Or “B,” is the workflow objects, variable, requirement, step, XML? Or “C,” requirement, variable, step, object, XML? Or “D,” object, requirement, variable, step, XML? We'll go ahead and look at the results here. The correct answer is “C,” and I just—I went through that very quickly just before we did this poll. We go through requirements. Once we know the requirements, we know the variables we want to use as inputs. We make test procedure steps. The test procedures

select the correct objects, and then the test procedure makes an XML file. So again, “A” would be incorrect.—XML is an output—it's not an input at the beginning. “B” is incorrect. You can't begin with objects until we know what the requirements are. And “D” is also incorrect. Objects are not known until requirements are known and test steps must be known before the objects are selected. And again, that was kind of detailed, but again, we went through it just before that. This is what the TPG does in order. So that's the way—we start with requirements and it traces all the way to our XML script that you can use to run the actual test procedures.

This brings us kind of to the end of Learning Objective 6, so let's just go through a summary. In this learning objective, we learned that the TPG follows NTCIP 8002, and we do that to ensure a uniform numbering system and content across all the 1200 series standards. We learned that the role of the TPG is to enforce the proper use of keywords as well as to create a test procedure in IEEE 829 with XML outputs that can imported into off-the-shelf automated test equipment. We also stepped through a successful installation of the TPG using the install wizard. And at the end, we used the TPG to develop an example test procedure for NTCIP 1203 dynamic message sign, a simple example. The TPG is following the content set by NTCIP 1207 for testing documentation, but slightly modified to meet 829 as well. We bring that up because we learned in the prior parts of this module and other modules that IEEE 829 is a—we're not going to go through it again, we're not really teaching it here, but that's what this is based on. So IEEE 829 goes through the concepts of level testing, master test reports, and we covered that so we're not going to cover it again. The point here is that NTCIP 8007 had to modified a little bit to meet 829, so there was a little bit of gap between the two but TPG has to follow the format set by NTCIP 8007. So it was modified a little bit so we could stay within the 829.

Moving on to Learning Objective Number 7. We'll now pick up and expand on a topic that we touched on in Part 1, and that question is: “How does the TPG and test procedure fit into a real-time system design and procurement?” So we're going to discuss the lessons learned from unsuccessful ITS projects and explain how to avoid repetition, and that's where we begin by playing the test procedure to be used successfully at the end. We don't want to go all the way through a project and at the end say, "Okay, now we're going to start testing it. Now we're going to try to figure out what the procedures are. Oh, I got unexpected results, and the person that designed it didn't know what was expected.” So the whole idea here is to do a lot of this work up front in the project. So we're going to end up with how to include clear and unambiguous contract terms and conditions and demonstrate how to work backwards from the project end to the beginning. We've touched on this in other modules but we're going through it again here to show that we're

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

kind of going through the whole workflow and we're getting—at the end—and we're actually writing our procedures in the testing workflow. So we want to go through again and show where that fits in.

So a quick refresher. Again, we went over this in Part 1. For successful project conclusion, don't approach an ITS project in the traditional—like you're doing a roads project or something that's not really related to high-tech and information technology. This is really high-tech, IT projects, so we really want to go through the testing workflow in that manner. So rather than approaching the beginning of a project as if it's ending as an IT project. Again, we got to start in the beginning knowing where we're going to end up, and what we do, or what I recommend doing is imagine when you're planning the project in the beginning imagine that you're the person at the end who's going to run the test and verify that the deliverable—like in this case, our DMS works correctly. So if I'm writing a contract, I want to say, "Maybe I should write this around if I was the person at the end and I wanted to verify it and I want no unexpected results. I want it to be right. What should I put in the contract so that everybody that's working on it is all using the TPG and they're all doing the same thing the same way and it's tested throughout the entire lifecycle instead of just at the end where you have unexpected problems?" So we want to know what tools and procedures we want to use, what pass/fail criteria. Let's think about all that upfront and include it in the contract, so that way the designers can use the process throughout the project to eliminate the unexpected results at the end. Because if they're eliminated at the end—or they go all the way through and they're at the end and you get unexpected results, that can be very expensive to fix because it could have been fixed in the beginning and not carried all the way through.

And again, this is kind of a refresher, but again, 829 includes level tests. So these are levels. So this is our concept of operations of our system, this is our traceable across, left to right on the dotted line. These are our levels. We might have a unit level, might be one piece of software, one piece of hardware, test it to make sure it's right. Subsystem level, put them together, make sure that this hardware plays well with the software. Go up another level: system verification deployment. Put them all together in a system. All integration, often called integration—that level is integrate the subsystems together, make sure it works. System validation, populate it with real data and make it work and do things that you expect it to do and make sure it works correctly. So if these levels are included in the contract, the TPG can then be used by the manufacturer throughout the software/hardware design and in installation. So the person that's down here, if I'm actually doing the hardware and software design, I would like to have all of these levels stated over here on the left side when I get my requirements and what I need to do to design. So the reason we're bringing this up again—maybe it's starting to get repetitious

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

throughout the module; but the TPG should be considered from the beginning. That's what we're saying here. We might want to say we're going to use the TPG in the beginning. The agency or the customer that's working on it, the manufacturer that's doing the work, we're all going to use the TPG, we'll all do it the same way, and we're going to test it throughout this whole lifecycle while we're designing and doing level testing.

Recall from Learning Objective 3 of Part 1, again, we said that—again, this is a repetition. We want to emphasize that again here. Remember that we saw that there was a study that showed that the odds are slim that ITS projects will be completed on time, within budget, and including all the user needs. So if you remember, we had a chart there that showed there was a real study done of IT projects and how many were actually successfully completed on time, under budget, without eliminating any of the requirements to keep it on time and on budget, and it was pretty dismal. What we're teaching here is if you follow this workflow and worry about all this stuff up front, even if we're spending 20 percent of our project in the planning stage, recall that that study showed that you'll be more successful than if you leave it all to the end.

And from the previous slide, we know ITS projects will likely fail unless we take a different approach, and that's what we're teaching here. Here we demonstrate an alternative method to avoid repetition and failure. Instead of designing everything as a one-off custom system for each deployment as if we were beginning a new project, like for doing a new product or a new road or something, what we're proposing here is to begin each ITS project from the viewpoint that you're at the end, like it's a real IT project, you're going to be delivering software, you're going to have communication protocols, that have standards. Let's plan the ending out in the beginning and include it in the contract. So each step for the system design, then each level has a test for that level, and we can make sure that we're testing it and tracing it all the way through.

And again, I think we showed this graph, we'll hit it again here. This left-most vertical axis represents the typical project stage. So we went through, in the B diagram, we went through concept of operations, requirements, design and construction, et cetera, operation and maintenance at the end. It's important to know. I got some of these slides from, it's in our references here at the end—System Engineering for Intelligent Transportation Systems on USDOT website. I encourage everybody to read at least the first 10 or 15 pages because it's really—if you've worked on projects for a while, you can recognize a lot of the problems that show up, and it'll tell you that these are complex projects and we're dealing with complex technology; things are going to go wrong. Things are going to happen. You just have to expect it. So defects are going to be inevitable, so

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

we don't want to write a contract to enforce the corrections of defects at the end, and that's been done before and a lot of us have worked on contracts that are like that, where it's not really clear what we're trying to do until the end. "Oh, here's your test procedure and—oh, got unexpected results." So we don't want to do that. We want to fix them when they're created. So if you fix it early, you can see that the cost to fix gets less and less and less, and I think we all agree with that.

So that's the reason we're using a Requirements Traceability Matrix. So it may be more costly in the beginning to plan what the Requirements Traceability Matrix looks like. It may take some more time to download the TPG. It may take some more time to populate all that stuff, like we did in our simple example. But you may be saving 100x the cost to correct the defect later, because you might be throwing away big portions of the completed design. So again, what we're teaching here is the workflow. Start with test in the beginning. USDOT has provided this TPG tool. It's going to help you out a lot, and that's what we want to do. We want to get away from anything unexpected. Everybody that's working on it, step by step, level by level, should know what the expectation is.

In previous courses, again, we learned to take a bottom-up approach. And again, this is all repetition, but we want to tie this all together since we're coming to the end of our workflow here for testing. And we want to extend this approach to the contract terms. So the contract should provide the information. It should require certain deliverables, and it should prohibit bad practices.

Some information includes the varieties of test equipment. It should include the applicable standard for each variety of equipment you're using. We should look at the conformance groups expected to be implemented, and test cases and expected test outputs. Those bullets shown here, that sounds like a lot, and it is a lot. All of that information has to be done some time, so we're saying the place to do it is upfront so that it can be used, and the tools can be used—the TPG can be used through the entire process, and the way to make sure that happens is write a contract that says, "This is the workflow. This is how we're going to do it. We're not going to pass the planning milestones until we see all of these pieces in place, and then we'll move on to the next stage of contract."

And again, we're recommending deliverables at the end should be in IEEE 829 format, and using the TPG to create XML files. And again, we talked about this before, it may be repetitious, but IEEE 829, if everyone uses that standard, and say you ran the test and you got an unexpected result, everybody's going to know that the unexpected result

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

needs more investigation; they're all going to do things the same way and the records are going to look the same, the formats are going to look the same. It eliminates a lot of meetings and emails and mistakes if everybody's doing it the same way and the documents all look the same.

We also said prohibit bad practices. That means you can put in your contract manufacturer-specific objects are banned from use if there is a standardized object that can do the same thing. And keep in mind manufacturer-specific objects—TPG is not going to do those. Block objects, the TPGs aren't going to do those. And remember why. We remember that those are specific to a manufacturer, so you will have to take into account block objects and MSOs and you'll have to have some extra documentation and write some more manual test procedures. I mean, you can do them in the TPG but you're going to have to figure them out yourself and you're going to have to get information from the manufacturers so that if my central is talking to someone else's message sign, and it uses something specific to me, I'm going to have to let the message sign manufacturer know it and we're going to have to agree on a test procedure. And you can do a lot to help on that when you're writing your contract terms. You can just say if there's standardized objects, you shall use those; we're not going to accept anything else. Or if you do, that's fine. I mean, if there's things in the design that you can't do as standardized object, that's fine, but if you can it's good to eliminate those. It makes your life easier.

We're on to another activity here. The question here is, "ITS project costs and schedules should be developed in which of the following order?" So you select from the four answers. "A" is from the test procedures back to the project workflow, or "B" is it from the requirements only? Or "C" is it developed to enforce contract terms at the end of the project, or "D" is it to minimize the planning costs up front? We'll go ahead and show the correct answer here. The correct answer is "A" proving the test procedures in the contract terms allows the manufacturers to use the TPG throughout the design phase to eliminate defects early. If a defect comes up, the person that's doing the design can test it and say, "Oops, it's wrong. I'll fix it," before it goes on to the next and ends up at the end in a system integration or a field test. So that makes "B" from requirements only, incorrect. Providing only the requirements allows multiple interpretations that could be costly to correct at installation. "C" is also incorrect. Contract terms should enforce continuous verification throughout the project, not enforcement at the end. And "D" is also incorrect. Additional up front planning reduces overall cost. And again, if you look at that system engineering guide, there's a good study in there that said, "You know, we're spending 15 percent up front, but that eliminated 15 percent in overhead."

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

To summarize Learning Objective 7, from Part 1 we know that the ITS projects have a low probability of success if planned and executed as a traditional roads project, or any type of project. We have to take this as a high-tech IT project, hardware and software. And the lessons learned from successful ITS project testing ensures success. Continuous testing throughout, from beginning to end, ensures success at the end. We learned how to avoid failure repetition by planning the project from the viewpoint of the project's end. We learned that by using clear and unambiguous contract terms and conditions we can provide test cases and test procedures as part of the contract for use by developers throughout the project. We learned that a good procurement process begins by answering the question, "What is success?" Let's don't get to the end and say, "We have four people that are arguing over what success is. Let's put that in the beginning." And then we document that success in the contract as test procedures and expected results. So again, that way all the project stakeholders can test throughout the entire design process.

Okay, we're onto Learning Objective 8 here. So we're going to take one more step here before the end of this second part of this module, and we're going to go through this pretty quickly, but just to give you a feel for something. We did a very simple DMS sign just to show you how the TPG works. Let's say we're going to fix some other things. So we're going to develop a complex test procedure that pulls together NTCIP elements by using TPG. So in this case, let's say we have a central system that's sitting there running. It's got some actuated signal controls out there connected to the central. Maybe it's a message sign that is running, and I'm the person that has to write a contract to expand the system. So we might have to add a variety of equipment to the existing system. So we may have to start by creating a set of manufacturer-specific objects. We're going to sort of pick a simple one here that we know the standardized objects don't cover it, and we're going to use the TPG as an acceptance test throughout, and we're going to explain how terms and conditions are based on the test procedure, and then the pass/fail criteria.

Let's assume we have to expand the existing system connected to the central station. We begin by taking an inventory of the requirements currently supported by the central station. Those requirements associated will automatically show up as part of the TPG operations that will be realized by manufacturer and optional objects for each of the equipment types at the roadside. So let's say we know what our requirements are. We take an inventory of what's out there. The system already meets all of these requirements. Now we're adding some more requirements. So those requirements, if we brought them into the TPG, they will be realized by manual and optional objects. So let's say that we go through that step and we brought in our standard, and we have some more requirements that we know can't be met by the objects of the standard. So these

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

leftover requirements can then be identified that might require unique design, and then finally we would need to determine that the existing communications media that is installed and the media performance in terms of speed and resiliency. So if you're upgrading a system, you're going to run into a lot of other things, other than just the standardized objects. Let's just go through this.

So say we collected the inventory of existing requirements that are currently supported. So then we would have to prepare a list of additional equipment required to meet the system requirements. So we're expanding the system. We have some new requirements. From this list, we will then identify the gap. So the gap between what the system does now, the new requirements we want it to do—the gap between it represents the content of the project we're planning.

So next we would identify any special requirements that cannot be realized by using standardized objects. So let's say there's a lot of work in adaptive these days, so I sort of picked out in this example, we might have a requirement to automatically adapt signal timing to abnormal traffic flows. So we don't want—if there's a university football game and it lets out and say 11 thousand cars normally went up and down the corridor on an arterial and then suddenly there's about that many coming from the side. We don't want to send officers out to direct traffic; we want the system to automatically look at the cars approaching and adjust the splits and offsets so that we can control wildly changing time bases in real-time. So if we know the approach speed limits, the signal time is required to be adjusted every three seconds to eliminate most of the stops and the least travel times. So the requirement is we know the speed limit, but the requirement is to adjust the signal timing every three seconds based on the approach speed.

So to make this to a successful conclusion as a project, we need to develop and include test procedures and include them as part of the contract terms. So for each set of requirements, we would use the TPG to create a test procedure header and then to select requirements to fulfill user need. So we can actually do the special requirement using the TPG. So we'd do an ID, we'd do a header, and select requirements, fill in the user need-to fulfill each user need—and then after the TPG creates the test procedure, we would then enter the variables. Next we'd define the steps of the test procedure and select the requirements from the list provided by the TPG, and in the end we would create and store the expected XML output. So we could do this for each one. But again, you can use the TPG. It won't automatically find anything in any of the 1200 series standards to do adaptive control, but you can certainly use the TPG to write a test

procedure for it and it'll automatically give you the fill-in-the-blanks and will put it in the correct format so your test procedure meets the format standards.

So we touched on this topic in Part 1. Before beginning a project, the contract should include the test procedure for each level. Levels should include the expected outputs so the design can be tested at the earliest possible point and the contract should ban the use of MSOs. But it could also—as we just went over, you might want to add or allow manufacturer-specific objects for some things that the 1200 standards don't cover, such as adaptive. So if you have special needs, the contract should say, "If it's just doing phasing, we're going to use standardized objects. If going to do adaptive, we're going to create some sort of an object or a dialog that can adjust the timing by stages or whatever, it can allow that, but it should very specifically and very critically identify those and delete the ones.

And then finally, the contract should require a test report at each level test that indicates the design produced at each level before proceeding to the next level. So your contract should say, "I know there's going to be levels. We're going to do this level, which is message sign objects and are maybe just a module, then we're going to integrate them into the system, field test, et cetera." There should be a procedure for each of those levels as we go up the right side of the "Vee" so that the expected things that are going to go wrong can be weeded out as we go along instead of at the end.

So again, we're going to put the test procedures in the contract in the beginning so everyone will be using the TPG throughout the system design. And again, we won't go over this in any detail—but again, the expected results aren't failures. We went through that. We have the anomaly report. The contract should say pass/fail report is pass equals pass. The expected failures requires an anomaly report for further investigation. The deliverable is an anomaly report for the results though the investigation.

So, again, successful project conclusion results and continuous testing. We've talked about that. But in the end, the TPG will develop the test scripts and will put those into automatic test tools.

Again, real quickly, 8007 sets the formats for all testing documentation. TPG uses the 8000 format that was modified slightly to meet 829. We talked about that, while 8000 sets the format for documentation, had to be modified a little bit, and we talked about that.

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

We've been looking at some of the questions that have come in, and one question that comes in on the chat box is: “What are the list of standards supported by the TPG?” So the list of standards supported by the TPG are the NTCIP 1200 series standards—NTCIP 12xx standards that have system engineering content. That means NTCIP 1200 standards that were done with a system engineering process. We went through the whole “Vee” model, just like we showed it, and the TPG supports those. That'll be 1203, 1204, message sign and environmental sensors, 1209 data elements definitions and transportation sensors,1210 field management station Part 1, 1211 objects for signal control and prioritization, and 1213 object definition for electrical management systems. So when we opened it up for questions that was one of the questions that came up. So that's kind of the answer. TPG supports the 12xx standards with SE content. And when you start using it, you'll see that. If you just play around with it a little bit and try to bring up the different standards, you'll see the ones that come up and the ones that are not there yet.

Another question we've gotten is, "Where can I get the TPG?” We talked about that in Part 1, but again, it's come up again during this session as a question. Copies of the TPG can be obtained at the link shown here, and that's standards.its.dot.gov/deploymentresources/tools. And again, that was developed by USDOT and that's on free distribution.

A couple of the questions that came in were: "Where can I get the TPG?", so that's shown, and then "Which ones are covered?” So we answered those questions. So we'll see if there are any more questions. It appears as though there aren't any at this point. So let's go ahead and go through what we've learned.

NTCIP 8002 will define uniform numbering and content for NTCIP 1200 series, center-to-field standards. We learned that using the TPG to enter inputs ensures the proper use of keywords and creates test procedures in IEEE 829 steps. We learned that TPG outputs in XML format provides test documentation and can be used as inputs to automated test equipment. We learned to begin by planning the test procedures to be used at the end. The contract should provide test procedures and expected outputs in order to verify requirements through the entire project. Contracts require objects and dialogs for special needs. We talked about that. And you should ban poor practices of undocumented manufacturer-specific objects for interoperability.

What we have here is our list of resources, and again you can see that we're listing as a resource T204 Part 1—you should have already taken that and the student supplement

T204, part 2 of 2: How to Develop Test Procedures for an ITS Standards-Based Test Plan

for this course. We go into that in a little bit more detail. If you pull up the student supplement for T204 Part 2 of 2, that's a good one. IEEE 829. Again, we didn't teach that. You just need to know where it is, and what it does is it defines the kind of the format. We did our—dynamic message signs is on the NTCIP website, as shown. That's where the modules are, module T201 and 202. And again, I'd encourage you to download System Engineering for Intelligent Transportation Systems and read at least the first part of it because that's got some very good information it. And again, test procedure in the TPG User manual. And you see in that bullet, that's where the person that is the contact at FHWA, and that is the link to where you can download it.

Questions. We answered two questions as we went along here, and seeing no others, we'll go ahead and end this session, and thank you for attending.

End of T204_Final.mp4