Module 56: Center-to-Center Reference Implementation: Applying the C2C Reference Implementation

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.

Ken Leonard: I'm 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 will 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. Thank you again for participating and we hope you find this module helpful.

Ken Vaughn: Hi, this is Ken Vaughn and we're here to talk about course T351 today. It's the Center-to-Center Reference Implementation: Applying the C2C Reference Implementation.

Ken Vaughn: I am your presenter today—Kenneth Vaughn. I'm the president of Trevilon LLC. I've developed test software to verify device conformance in NTCIP and also I've done testing services for the industry for over a decade. I'm well experienced with these testing issues and I'll be discussing these topics with you today.

Ken Vaughn: The course has four learning objectives. The first learning objective is learning how to install and configure the C2C RI on a PC. The second one will discuss how to operate the C2C RI software itself. Then we'll go on and retrieve the C2C RI results from a sample test. Then finally we'll prepare a report based on the C2C RI results. From these learning objectives—this is kind of the advanced course. It's actually how to use the software, so we recommend this for users who really want to get into the details and have quite a bit of experience in understanding how center-to-center communication works.

Ken Vaughn: With that, we'll go ahead and discuss a little bit about Learning Objective 1—installing and configuring the C2C RI on a personal computer.

Ken Vaughn: The first thing you have to do is obtain the software. The good news is that you can obtain the software for free from the government website at https://www.standards.its.dot.gov/DeploymentResources/Tools. Once again, that software is free and it's updated periodically with the latest version. That download will actually include the user manual for it, as well. That user manual in some areas goes a little bit deeper than this course does, but this course provides several different videos that will assist you in understanding how the system really works. Also, if you need to—while you're using the software—you can obtain technical support via email using the email address of c2crisupport@transcore.com.

Ken Vaughn: In order to run the software that you download, you'll need to install it on a machine. That machine will have to have certain system requirements that we'll discuss. There's also some key interoperability requirements. Then we'll discuss the skill set requirements for the operator him- or herself.

Ken Vaughn: The first thing we need to look at are the minimum system requirements. We do recommend either Windows 7 or 8. We've not tested it on Windows 10 officially. I've run it myself on Windows 10 and it seemed to work reasonably well—but it's only been officially tested on Windows 7 or 8. This is the 64-bit Professional version of these operating systems. We also recommend a 2 GHz processor, 4 GB of RAM, 1 GB of storage, 1 Gbps Ethernet connection, and finally, the Java SE Runtime Environment JRE V7.17.

Ken Vaughn: The purpose of the software is to ensure that your systems are interoperable. What precisely does interoperability mean? It's the main purpose of ITS standards but what really do we mean by that? IEEE has a good definition. It is: "The ability of two or more systems or components to exchange information and use the information that has been exchanged." As a simple example, consider two systems exchanging comma-separated values—CSV type files. If the receiving system is a simple cloud storage system and does not know how to read the CSV file, the two systems could not be considered interoperable because the receiver is unable to use the information. But interoperability is realized when another system is able to read and use the CSV file, connects to the cloud service, and downloads the file. The cloud storage system still offers useful services that are not interoperable, but you don't gain interoperability until you have another system that actually can use that information. That's what we mean by interoperability—it's not only exchanging the information but you actually have application logic that can use that information. That's what we try to do somewhat with the C2C RI and certainly with the standards within ITS.

Ken Vaughn: What are the interoperability requirements? C2C interoperability testing requires several different things. In order to realize the desired level of interoperability, we must define our needs and specify the desired system with requirements. In other words, in our previous example, does the receiving system just parse the CSV file and add it to its database, or is the receiving system supposed to perform an operation based on the information? Needs and requirements define precisely what result is expected from interoperability, and the requirements claim conformance to various features. Standardized needs and requirements are defined in the TMDD. Modules A321a and A321b provide guidance on how to specify optional features from these standards. We also need a precise definition of the details used to satisfy those requirements. That's provided in the TMDD and NTCIP 2306 standards. They provide a precise design for each requirement, along with traceability back to the standardized requirements and user needs. But simply specifying how it works isn't really far enough. We need to go further and identify how we verify and validate the system then that verification and validation needs to be performed. Module T321 defines that process. The C2C RI extends this through Modules T251 and T351 in identifying how the software tool assists in that verification stage. It should be noted that the C2C RI focuses on verification; validation is something you do on the end system. One of the key things that we need to remember in the C2C RI and testing in general—we provide essentially spot checks. The number of possibilities are almost endless with this testing. We can't perform exhaustive testing—it'd simply require too much time. The C2C RI is currently designed to check the most obvious issues; the tool will likely be enhanced over time based on user feedback. Finally, passing C2C RI does not guarantee full interoperability because of the fact that we're only testing what we can in a reasonable amount of time. It does require user interaction at points, and it is not exhaustive. Finally, standards themselves may have ambiguities, and so an error within the testing of the C2C RI might reflect some other issue. Every issue identified by the tool needs to be thoroughly investigated to identify where that problem—what portion of

the system is responsible for that problem. Is it the standard? Is it the C2C RI tool itself? Or is it the system under test, a user error, etcetera?

Ken Vaughn: With that—and in order to do that sort of analysis—the operator of the test software needs to have quite a bit of experience and knowledge. We'll go over these in detail in Module T251, but we summarize them here. The user needs to have a background in encoding languages. The user needs to be familiar with things like XML, because the user will need to actually create XML documents and interpret the message content in many of these tests. The user also needs to be familiar with protocols—to understand what the tests are doing and to resolve problems in the test setup. This also helps them in debugging the process, when looking at logs and understanding what went wrong, when. Next, the user needs to be familiar with Windows networking and be able to properly configure the test to ensure that the message is being sent across the network. Then, of course, he has to be familiar with the ITS standards and know which requirements to check, and to verify that failures detected by the tool are in fact issues of nonconformance. This is a fairly advanced set of tests that are being performed, so the user really should have some testing experience and know how to perform repeatable tests in a methodical manner, according to industry standards; and to document that process along the way. The scripting language is perhaps somewhat optional. This is only necessary if the user plans to test things that are not part of the standardized features that have already been implemented within the C2C RI. But if he wants to extend it to project-specific extensions, he would need to expand the tool. To do that he would have to write his own scripts according to the process. We don't discuss that a lot in this course; there is guidance on how to write those scripting scripts within the user documentation. The final step is understanding the system under test. You need to be able to configure that system under test—to make sure it's ready to receive messages—and when things don't behave as expected, know how to work with that system to determine whether it's actually an error or if it's just misconfigured. Once again, a full description of each of these is in Module T251, but if you're taking this advanced level course, we assume that you have these skills.

Ken Vaughn: At this point we have downloaded the software and we've installed it on one machine. The next question becomes: how do we actually connect our systems? The C2C RI tool is designed to connect and test center-to-center communications and to make sure that they're conformant with our industry standards. To understand how to use this tool, we first have to understand some of the base terminology concepts involved in C2C communications. With any exchange in ITS center-to-center communications, one center is called the Owner Center and the other center is called the External Center. The Owner Center is generally the one providing ITS information to an external requester. The process used for the two systems to exchange data involves two steps. The first step is to discover the services that the Owner Center provides. The second is to exchange the data using those services. The discovery is archived by the Owner Center providing a file that defines the services that it offers using an open format. Using our deployed environment as the base condition, this graphic depicts how the C2C RI can serve as the Owner Center or the External Center. The C2C RI is configured for one mode and used to test the other mode. In this case—what's shown on the screen right now—you see the C2C RI acting as an External Center. It could be used to test an Owner Center, but it can be used the other way around. The point here is that the C2C RI would replace one end of the system. When you do this, it is simulated testing—you're actually removing one of the real systems and replacing it with the C2C RI.

Ken Vaughn: In our presentation, we're going to be demonstrating both modes of the C2C RI—then we'll use it to test it against itself in the other mode. We actually have two instances of the software running. This is a useful way to test your setup to ensure that you have the C2C RI configured correctly. Nonetheless, while our demonstrations will test between two C2C RI implementations, realize that a real test would only replace one of the end systems.

Ken Vaughn: With that, we come to our first pop quiz.

Ken Vaughn: The question here is: Which of the following statements is untrue—or not true? C2C standards define how services can be discovered; a C2C exchange occurs between an Owner Center and an External Center; option C) the C2C RI software is available for free; or option D) ITS standards only define how to exchange information. Go ahead and make your selection and we'll review the answers.

Ken Vaughn: The correct answer is in fact answer D—as in dog. ITS standards also define not only how to exchange information but how to use the information once it's been exchanged. C2C standards do define how services can be discovered. This is done through a WSDL file—which we'll discuss here in a second. It allows an External Center to discover the services offered by an Owner Center. The C2C exchange does occur between an Owner Center and an External Center. Those are just names that have been assigned to the two different systems, according to the ITS standards. The C2C RI software is, in fact, available for free. It can be downloaded from the web address shown here.

Ken Vaughn: To set up the C2C RI, there's a couple of steps involved. One is you have to install the C2C RI on your computer, and then define some of the initial configuration of that C2C RI. Once the C2C RI is properly installed, we will then configure it to support the services offered by the Owner Center through the use of a service announcement file. Finally, we'll identify test cases that will be performed in our full deployment through the configuration.

Ken Vaughn: The C2C RI is distributed as a standard Windows Installer. When running this Installer, it is strongly recommended you install the program into its default directory, to avoid any erroneous error warnings when running the software. There are several different directories you have to configure. You can do that after the fact, and we'll show you how to do that. To minimize any problems, we recommend installing the default directory. Once the software is installed, you'll need to configure the default directories for file storage, along with some other basic information.

Ken Vaughn: This video shows that process of setting up some of the initial directories. You can go ahead and press Play on the video and you'll see how that process actually works in real-time.
<<start video>> In this video we'll learn how to customize the default directories and some other parameters for the C2C RI. To customize these values, go to Tools/Options, which will bring up the dialogue box for your options. You'll be able to configure your user name, as well as your organization name, and configure all of your directory structures. Any invalid directories will appear in red. If you need to, replace them in straight text or using the handy structure browser button. Also make sure to go to—this is on the general tab—make sure you go to the logging tab as well, there's another directory structure there. The other values should default to the correct values for normal testing. Once you're done, you can click on Save, and then Done.
<<resume module>> It is important to remember that all directories must exist on the local machine. Changing the values is especially important if the C2C RI is not installed in the default location. That's why we recommend putting it into the default location. If the default directories don't exist on the machine, you'll probably get some errors and they may not be completely informative. Just make sure directories are set up correctly.

Ken Vaughn: Now that we've installed the software, we can investigate how to use it. We mentioned that the first part of a connection between two Centers is the discovery of services. Services are defined in what's called the Web Services Description Language—or WSDL file. The WSDL file provides a definition of the services and messages supported by the Owner Center. It may also identify publication services that should be supported by External Centers that use subscription service. In other words, an External

Center would access a subscription service—but in order to access the subscription service, the External Center should support a publication service that the Owner Center would then send messages to later on. Discovery is achieved by the External Center retrieving the WSDL file from the Owner Center—further requirements of NTCIP 2306. It is the responsibility of the Owner Center to define its own WSDL file that conforms to 2306, and to make it available to authorized External Centers. Implementers should be aware that several off-the-shelf tools that assist in the generation of WSDL files do not produce files that are conformant to some of the rules in the current version of 2306. It is the responsibility of the implementer to overcome these obstacles. There's no restriction on the number of WSDL files a Center publishes. Thus, an implementation can publish both the generated version and the 2306 version, if needed. In short, it is the responsibility of the Owner Center to advertise the services that it offers. It does this through the use of the WSDL file. An External Center only needs to read this file and determine which services it wishes to use. The WSDL file format is based on XML. Thus, these documents are relatively easy to read—but you have to understand how that structure works. If you do, you can then use any number of off-the-shelf text editors to edit these documents. The primary purpose of the WSDL is to define the supported services of the Owner system. The primary purpose is to define the supported services to be used in real time. The WSDL file will also be used—because it's there, it will also be used by the C2C Reference Implementation as part of testing; but the primary purpose is to define the supported services.

Ken Vaughn: The WSDL file defines services in a standardized format. The WSDL file starts with references to the TPD and other schema files that define the standardized—as well as any customized—detailed data structures. The file then combines these data structures into messages—which are also standardized by the TPD—and then defines how the messages can be ordered to form dialogs—known as operations. This whole process is defined by the TPD. The WSDL file then associates the operations with the ports and binds them to specific protocol stacks. This is standardized to a small number of options. Finally, the file maps the bindings to services which are assigned a specific URL. The URL is specific to a deployment. The good news is that most of this information is defined in the standards. If you're using a strictly standardized interface, then you have very little to do. But that last step—defining the URL—that is specific to your project. There's no way for the standards to know your specific URL; you'll need to go in and customize that URL to your particular project. Further, if you want to offer services that aren't standardized, in addition to the standardized services, you would need to define and customize the rest of the file.

Ken Vaughn: WSDL files are associated with the Owner Center. The Owner Center is responsible for defining the WSDL file and storing it on its website—or in some location where an External Center can retrieve it. If you're operating the C2C RI in External Center mode, it would just need to configure the C2C RI to reference the WSDL file provided by the Owner Center—i.e., the system under test. In this case, the C2C RI will check the provided WSDL file for conformance to the standards. Some deployments have discovered that the WSDL files automatically generated by some tools do not conform precisely to the NTCIP 2306 format. The C2C RI will flag these issues, so that the developer of the Owner Center can correct them. If you're using the C2C RI in Owner Center mode, you'll need to provide the WSDL file for the External Centers to reference. Luckily, the C2C RI includes sample WSDL files, but these files will need to be updated to reflect the URL currently in use by the C2C RI Owner Center. For example, you'll need to change the default web address to reflect the IP address, or domain name, of your C2C RI instance. This is done in the last few lines of the file—we'll show you a video of that in a second. In fact, that'll be the next slide. It's done for you in one sample file, but be aware that the C2C RI relies upon several different instances of WSDL files for different tests. You'll need to change all of these files that you use in order for your test to work properly.

Ken Vaughn: <<start video>> You can open the WSDL file in any standard text editor. The WSDL file tends to be rather large—this is actually a fairly simple one, showing only a few messages—but if you scroll to the very bottom of the file, you'll find a section External Center bindings and Owner Center Services and, finally, External Center Services. The portion you need to change is the web address. Here, we see that the Owner Center Services has a location of http://C2CRIOC—that is the address you need to change—change it to your IP address. In my case, the address is 10.211.55.1. We just replace that value straight down to all the occurrences that you see in that file. In the case of the External Center, you actually don't need to change that value, because the Owner Center does not know the address of the External Center—that is provided during run time. So you only need to change the Owner Center values. Once you've done that, you can save your values, and then you're done. <<resume module>>

Ken Vaughn: We've now saved and created our WSDL file, but now we need to go ahead and configure the C2C RI to actually use those WSDL files. This video will discuss how you do that.
<<start video>> Now that we've saved our updated WSDL file, the next step is to link the C2C RI configuration file to the WSDL file. To do this, we'll open and edit our example configuration file. This example is based on the NTCIP 2306 test WSDL file, so we'll open the NTCIP2306OCS configuration file and the C2C RI will operate as an Owner Center. We then select the System Under Test tab and make sure that the web services URL address is appropriately set. Since we are acting as the Owner Center, the WSDL file we just edited is stored locally on our machine. The address starts with file://localhost/c:/ followed by the path and the file name. If we were editing the External Center configuration file, we would either need to reference the URL address of the WSDL file or the Owner Center server—for example, with an http address—or we would need to download a copy of that file to our local machine and then reference it with a local URL address that would start similarly with a file://localhost address. Note we do not need to worry about the IP address, port, or host name fields. This is because the WSDL file overrides these values. Right now, these fields have no effect on the current version of the C2C RI. Once the web service URL is properly configured, you can click the Close button and save the file at the prompt. <<resume module>>

Ken Vaughn: Once you've configured the C2C RI to use the appropriate WSDL files, the next step is to go in and configure the C2C RI to conform to the needs and requirements that you have for your specific project. You should have already done this through a PRL as part of your procurement process. Whatever protocol requirements list you used or needs-to-requirements list you used to procure your system, this is where you'll enter that information—it's almost a one-to-one correspondence. This video will show you specifically how you do that.
<<start video>> The next step is to configure the C2C RI for a specific test. To do this, open the configuration file and go to the appropriate Layer Parameters tab. In our example, we're testing the lower layers, so we'll go to the Application Layer Parameters tab. Here we can indicate which standardized options we want the C2C RI to test. The settings on these tabs will limit which test cases are visible, once you're in the C2C RI execute mode. The table structure follows the needs-to-requirements traceability matrix defined in the standards and should be filled out based on the options supported by your system under test, to ensure that you can run all applicable tests once you're in the execute mode. The upper portion of the screen indicates the user needs defined in the standard. Any items that the user defines as mandatory will be flagged for testing without any option to remove the flag. Users may flag optional needs—such as these—as desired. In this example there are three options, and we have flagged all three for testing. Your configuration should reflect the capabilities of the system you are testing. Selecting one of the rows will automatically populate the requirements table. This table works in a similar manner as above: items that are mandatory will always be flagged; optional items can be flagged as needed. The test parameters table is provided with the standardized needs to requirement traceability matrix and has

additional parameters defined for a requirement. This does not occur within the NTCIP 2306 but, if you look at the Information Layer Parameters tab, you can see how this works. If we select the Verify Connection Active user need and then the Send Center Active Verification requirement, the test parameters table is populated with one entry that describes the required response time for the Owner Center. The test procedure uses this value to determine how long to wait for a response, prior to failing the response step. As with all other entries on the screen, the value assigned for this parameter should be based on the advertised capabilities of the system under test. Once you are done with your selections, you can select the close button and save your settings at the prompt. <<resume module>>

Ken Vaughn: Now you're fully configured and ready to test. We've discussed how to install and configure the C2C RI and have customized the WSDL file. We then configured the C2C RI to use that WSDL file, and we configured the C2C RI to use the needs-to-requirements that you've defined for your project. The next step will be actually running tests.

Ken Vaughn: Before we do that, we have our second pop quiz.

Ken Vaughn: What is the primary purpose of a WSDL file? A) Configure the C2C RI; B) Report errors to the user; C) Define services offered by the Owner Center; or D) All of the above. Go ahead and make your answer.

Ken Vaughn: The answer is C. The WSDL file defines the services offered by the Owner Center and how remote systems can access these services. While it is used for configuring the C2C RI, the purpose of the file is much broader and includes operational deployment. It does not, in fact, report errors to the user—and because it does not report errors to the user, "All of the above" is also an incorrect option.

Ken Vaughn: That completes our first learning objective—installing and configuring the C2C RI on a PC. The second learning objective is actually operating the C2C RI.

Ken Vaughn: We'll learn how to actually operate it and understand how to interface with it.

Ken Vaughn: The first video that we'll show here is designed to show you how to actually execute a particular test. In the first segment, we defined how you configure the C2C RI to reflect the needs and requirements of your particular project. Based on those needs and requirements, the system will automatically identify which tests might be appropriate to your particular project. It is then up to you to identify which tests you want to perform and then actually go through the process of performing those tests. This step will show you how to start a test on an Owner Center, including how you get past all of the initial prompt screens.
<<start video>> To execute a test, select the Execute option from the File menu or select the Execute button. The C2C RI will prompt you for a test name. In this case, we will enter OC—for Owner Center—Test and select the configuration file. The configuration file we selected before was NTCIP2306OCS. After clicking on Accept, the C2C RI will enter its Execute mode. Since we are testing for lower layers, click on the Application tab in the left hand window and then select an appropriate test case. You can see a description of each test by hovering over the test name. We'll select the first test case, which is SHRR-EC. This test will test to see if the External Center—EC—can communicate using SOAP over HTTP SH for request-reply exchanges—RR. We can now run our test by pressing the Run button. Many of the C2C RI test cases include variables that can be configured by the user. These variables can be set in the data files or can be presented in prompts at the start of tests. By default, this test prompts the user for several values. The first value is the name of the service defined in the WSDL file that the tester wants to use for this test. Remember, this test is only testing NTCIP 2306—i.e., the lower layers of the communication stack. But to test these layers, the C2C RI needs to sense something in the upper layers. It doesn't matter what we send, as long as the External Center supports the information and conforms to the parameters of

this test. In other words, in this case it should be a request-reply SOAP over HTTP service, since we chose the SHRR test case. We can find the service name by looking at the WSDL file. Back where we entered the address for Owner Center, we discover the name of the service we were specifying was tmddOCSoapHttpService. We can copy this name into the prompt. The C2C RI then prompts us for a subscription port name for Iteration 1; once again, this is defined in the WSDL file. The tmddOCSoapHttpService includes several ports; the one for SOAP over HTTP is called tmddOCSoapHttpServicePort. We will copy this name into the prompt. The C2C RI then prompts us for the subscription port for Iteration 1; once again, this is defined in the WSDL file. In this case, we have to realize that the tmddOCSoapHttpServicePort is of type tmddOCSoapHttpServiceBinding. If we search the WSDL file for this name, we discover it is defined for a variety of operations. We'll pick the OP_DMSInventoryRequest. Finally, the C2C RI prompts for the response message for Iteration 1. This is the actual instance of the XML message the C2C RI Owner Center will respond with, when an External Center makes a request. The C2C RI does not generate this automatically,--and this is a key reason why the tester must be very familiar with XML and the TPD standards. However, the WSDL file does reference the schema that is a template for this message. If we search on the operation name, we discover that the output message—i.e., the message to be provided by the Owner Center—is the message tns inventory message. If we search on that term, we discover that this message consists of a single part, as defined by TPD's dMSInventoryMsg. We then just have to develop an instance of an XML message, using TPD standard format, and paste the resulting string into this prompt. Luckily, I already have a sample message saved, that I will copy and paste as appropriate. The Owner Center instance of the C2C RI now has all the necessary information to perform the test. <<resume module>>

Ken Vaughn: The last video described how you configure a system and start a test for the Owner Center. This slide will do the same thing, but it will be for the External Center. An Owner Center communicates with an External Center. The C2C RI can perform either role. In the last slide we talked about the Owner Center—in this slide we'll talk about the External Center.
<<start video>> To run the test, we also need to make sure that our External Center is running. To do this, we'll launch a second instance of the C2C RI. We follow the same process to start this instance, but we will use a configuration file set to run in External Center mode, rather than the Owner Center mode. Because we are running as an External Center, we end up with a slightly different list of tests in our Application tab. To make sure that both instances expect the right sequence of messages, we need to make sure that we choose a complementary test case in the External Center. In this case, it is the SHRR-OC test case. We then press the Run button, as we did before, and answer the prompts. The first three prompts will be answered the same as before. In other words, the External Center should call the operation using the port and service that the Owner Center was configured to expect. However, the fourth prompt will have different response. Whereas the Owner Center was configured with the response message, the External Center will send the request message. The format of this message is also referenced in the WSDL file but is the input flow to the Op_Inventory list. Luckily I also have a copy of this message stored locally. Now both instances of the C2C RI are ready. <<resume module>>

Ken Vaughn: We now have both an Owner Center and an External Center configured and ready to go. We understand how they both work. This slide will actually show the two Centers working side-by-side exchanging data and shows you how the system produces results.
<<start video>> Now both C2C RI instances are running side-by-side with all the information that they need. The instance on the left is the Owner Center; the instance on the right is the External Center. It is the tester's job to make sure that the two systems are ready for the exchanges to be tested. In the request-response model that we are using here, the Owner Center listens for incoming requests and responds when requests are received. As such, we need to make sure that the Owner Center is listening

before the External Center transmits its request. To do this, we'll accept our final input value on the Owner Center first. After some initial checks, the C2C RI will indicate that it's ready to receive an incoming request. We'll accept that message and then accept the final prompt on the External Center. The External Center instance then performs various checks and eventually sends the request message to the Owner Center. The Owner Center instance then responds automatically, and both instances process the information that was exchanged. The final results are recorded at the top of the tables and more detailed information at the lower tables. Once testing is complete, the user can close the window and the file will be saved automatically. <<resume module>>

Ken Vaughn: You may have already noticed that entering variables at the prompts—especially entire XML messages—can become quite time-consuming and prone to error. Any error in those entries can cause a test case to fail. The prompts provide little screen space to double-check what you've entered for the XML message. In order to overcome this, the C2C RI tool allows the user to provide this information in a data file rather than directly in user prompts. This allows the user to enter the data once and then perform the test multiple times. The User Guide describes how to create a data file from scratch. However, the easiest way to create these files is to go in and edit the default data files that are hidden within the tool itself. We'll investigate how to do that in our next video.

Ken Vaughn: This video will show you how to obtain a copy of those hidden data files, so you can go in and simply modify those as needed.
<<start video>> You create your custom C2C data file, using any text editor, but the easiest way to make sure you define all the needed parameters for the test case is to copy the default data files provided by the C2C RI itself. To do this, you'll need to extract the appropriate data file from the jar archive. This can be done with any file archiving software package that supports jar files. We'll use the 7-Zip file manager. Open 7-Zip and navigate to the C2CRI navigation directory. Then go to the Test Suites subdirectory. This directory contains several files. Open the jar file that corresponds to your test case. In our case, we're interested in the NTCIP 2306 test suite, so we open that jar file. Inside of the jar file, open the appropriate layer directory. In our case, we have an AppLayer directory (the TMDD jar file uses an InfoLayer directory). Open that directory, and then inside of that, open the Data directory. You will now see a list of every test case defined in the test suite. Many of these tests were designed to test the C2C RI itself, and will only appear in the C2C RI user interface if you use some advanced options. All that you need to be concerned about are the file names that correspond with test cases that you want to customize. In our case, we'll find these at the very bottom of the list: TCS-C2CRI-NTCIP2306-WSME-SUT-SHRR, and we want both the EC and the OC files. Now we need to save these files to our desired location in an unencrypted format. 7-Zip allows us to do this by right-clicking on the files and selecting the Copy To option. We can then select the location we want to save to. These files are now stored in that location; we can open them with any text editor. This is the default data file provided by the C2C RI. All we need to do is to edit the tags that say USERDEFINED to the actual values we would like to use. In this case, we use the same values that we entered at the prompts before. Now we'll save the file. We also make similar changes to the OC data file. The final task is to modify the C2C RI configuration files to use with the supplied data files. To do this, open the C2C RI configuration file; we'll start with the EC file. Select the Layer Test Cases tab—in this case, we're dealing with the Application Layer, so we'll select the Application Layer Test Cases tab. Then go to the selected test case. Our test case is near the bottom of the list: we're interested in the TCS-C2CRI-NTCIP2306-WSME-SUT-SHRR-OC and -EC test scripts. We'll set the source of these to the file names of our data files. Once we've customized our data—once we've customized the settings, we can close the file and save our data. We'll repeat this with the OC data configuration file. Now we can launch our two instances of the C2C RI again; we'll start the OC first, so we'll start listening. This time, rather than prompting us for information, the tool immediately begins to

perform checks and lets us know when it's ready to receive. We'll accept that notice and then begin the External Center instance. Likewise, it immediately begins its checks without any prompts. This level of automation is critical for reliably repeating tests in an efficient manner. <<resume module>>

Ken Vaughn: As shown in the video, the tester still needs to know how to read, edit, and create both WSDL and XML files. However, data files allow the tester to prepare this information once for each scenario to be tested—and then streamline the testing process over and over again, if needed. Within Version 1 of the C2C RI, the response message configured in the Owner Center must be the exact message that you want sent to the External Center as part of your testing process. However, in Version 2, the XML message sent back can—the configuration file can contain a superset of the information that you want sent back. Version 2 will enable filtering of that message to meet the needs of a particular request. For example, if the request is for assigned inventory within a certain geographical area, the Owner Center can be configured with a message that contains signs in a larger region. When the request comes in for that small area, the C2C RI in Version 2 will subset that response message and send back only those signs that are in the requested area. In Version 1, it would send back the entire message—whatever that entire message is. If you show the entire region, it would send back all of the signs—which isn't really technically true, because the message request is asking for that large list to be filtered. Version 2 will include that filtering logic.

39

Ken Vaughn: It's also worth noting that the C2C RI supports sending multiple publications to a single subscription message. For example, I can configure my External Center to request it to subscribe for information about events. I can then configure my Owner Center copy of the C2C RI to support multiple events reported at different times. I can have one publication sending out that shows a particular event. At some point later, a second publication going out to that same original subscription showing a second event. The data file for a subscription publication test case can be written to define different variable values. Each of those two messages sent from the Owner Center to the External Center can, in fact, be completely different. The user can define those. That's done by using this Iteration tag. You'd say #ITERATION NAME = One; then give all the parameters you need for that particular iteration. Those are essentially the prompts you would receive if you didn't have a data file. If you wanted to have a second publication, you would simply have a new line that says #ITERATION NAME = Two. You'd then give those same variables whatever values you want. That allows multiple publications for a single subscription.

Ken Vaughn: That brings us to our third pop quiz.

Ken Vaughn: How can you get the C2C RI to publish multiple publications for one subscription? A) Enter the number of publications in the configuration file; B) Use the #ITERATION keyword in the associated data file; C) Multiple publications are not supported by the C2C RI; or D) The C2C RI can receive but not publish multiple publications.

Ken Vaughn: The correct answer is B—as in "boy." The user uses the #ITERATION keyword in the data file. A is incorrect because the configuration file does not have a parameter like this—the number of publications parameter. C is incorrect because the C2C RI does support multiple publications using the #ITERATION keyword. And D is incorrect because the C2C RI is able to both send and receive multiple publications.

Ken Vaughn: That completes Learning Objective 2. We've talked about installing and configuring the C2C RI on a PC. We've also talked about operating it.

Ken Vaughn: Now we'll talk about retrieving the test results from a test.

Ken Vaughn: The C2C RI presents results in real-time on the screen. The upper pane provides a summary for each test performed and the lower pane shows the step-by-step details of the selected test. When a step fails, the Results cell provides a description of that failure. This allows the user to diagnose the problem without having to go through the entire process of generating a report. The downside is that the screen space is limited and the report format—which we'll discuss later—may offer an easier way to read the information.

Ken Vaughn: It should be noted that when the failures occur, they may be due to a variety of different reasons. Other than the system under test having an error, there are a variety of different possibilities. While that is one potential source of the problem, the problem could also be due to a configuration error, a network configuration error, a user error, a C2C RI configuration error, an implementation error, or an ambiguity in the standard. Each failure reported by the C2C RI must be investigated in detail to determine the cause of the failure. What we don't want to do is to blame the implementation for everything when, in fact, it may have been just our own user error.

Ken Vaughn: As an example, let's investigate the error reported on the previous screen. This says that the test failed due to a transport error result. In other words, an error occurred related to the transport layer of the communications—in our case, the TCP connection. Let's consider what the likely sources are for this error.

Ken Vaughn: The TCP stack tends to be off-the-shelf software that is used for virtually any communication task. It's unlikely that the error resides in the implementation of either the SUT or the C2C RI. Further, the standard has been well proven over the years, so it's unlikely that the problem is due to an ambiguity in that standard. Finally, it is unlikely that any user error in performing the test would result in such a low-level error. That only leaves three potential errors that are the ones most likely to be the problem. One would be the improper system under test configuration—such as having the system under test listen to the wrong port number, so it never hears the message coming in. Another possibility is the improper network configuration. By this we mean something along the lines of the network is configured in a way that prevents the message from properly being routed. If I'm going through a complex network, maybe one of my routers is misconfigured. A third possibility is an improper C2C RI configuration error—such as sending to the wrong IP address or port number. Any one of those three would be the most likely sources for your problem. This is almost certainly a configuration error rather than an error with the system under test. That's the process you go through for every error identified. Identify what types of errors are unlikely; from those you can identify which ones are more likely, and then you can investigate each one of those.

Ken Vaughn: The user can then diagnose the issue by simply following the logical flow of information. The C2C RI logs can assist in isolating the error. The Message Summary report will identify the messages that are sent and received by the C2C RI. There's a good chance that the system under test will have a similar log capability. So one should be able to check to see if one system attempted to send a message, while the intended recipient never received it. However, the final resolution of the problem may need to use a line analyzer to identify why one system believes it is sending a message, while the other system does not record it. For example, maybe the sending system was never able to actually transmit on the wire. Maybe it was using a different communications facility, so it never hit the wire that the other machine was listening to. Or, conversely, you discover that the message did go across that wire. The receiving system software may have received it at a very low level, but for some reason rejected the message because maybe it detected a bit error or something else. The same sort of process of following the logical flow of what should be happening should assist in identifying the source of the problem.

Ken Vaughn: All issues should be documented in anomaly reports. These reports follow the format defined in IEEE 829-2008 and should provide enough information that the issue can be recreated. Once the anomaly is identified, it should be resolved and the system should be retested. A user may wish to test all known failures before starting a retest of the entire system. But as software changes can sometimes have unintended consequences, it is critical to retest all test cases using the final version of the software.

Ken Vaughn: During the test, the C2C RI will automatically record all of the test results. Once the testing session is complete and the user closes the file, the C2C RI will encrypt and sign the information in order to securely store the results. The location of the resultant file can be specified as a part of the directory options under the Tools menu. As part of the process, the test is started, or after testing—the location of the file is either defaulted to the default directory, or the user can move it after the test.

Ken Vaughn: That brings us to our fourth pop quiz.

Ken Vaughn: Which of the following might cause the C2C RI to report a failure? An ambiguity in the standard; the system under test configuration error; the C2C RI configuration error; or D, all of the above.

Ken Vaughn: The correct answer is D—all of the above. All three options can cause the C2C RI to report a failure. A is incorrect because an ambiguity in the standard can cause a failure—but so can the other options. In the case of B, the SUT configuration error can obviously cause a failure—so if the system under test is misconfigured, that can also cause an error. And likewise, if the C2C RI is misconfigured, it can also result in an error being reported.

Ken Vaughn: That completes three of our learning objectives. We've installed and configured the C2C RI. We've operated the C2C RI. We then retrieved the test results.

Ken Vaughn: We'll now move on to our fourth learning objective—preparing a report based on the C2C RI test results.

Ken Vaughn: This video will describe how you prepare the reports from within the C2C RI. Now, these are not IEEE 829 reports, but they are the reports that will be generated from the tool that will allow you then to create your own IEEE 829-2008 reports.
<<start video>> The C2C RI offers the user the ability to produce a variety of different reports. Any of these reports can be generated by selecting the Reports option under the File menu. Once the dialogue box opens, select whether you'd like to generate a configuration report or a log report, by selecting the appropriate tab. Once on the tab, you can enter the desired report file name and select the configuration or log file you want to generate the report from. Finally, select the type of report you'd like to generate, and press the Create PDF button. The C2C RI will then generate a pdf file, as directed. Once the report generation is complete, you can go to the directory structure and open the pdf file with any pdf reader.
<<resume module>>

Ken Vaughn: It should be noted that the C2C RI does not directly generate standardized IEEE 829-2008 reports, primarily because it simply does not have all of the information that it needs. Instead, the C2C RI generates reports that assist users in developing their own IEEE 829 reports. This table maps the IEEE reports to the C2C RI reports that assist in their creation. We'll discuss each of the IEEE 829 reports on the following slides. You see here—Test Log—you might need several different reports, because they all log information. The Anomaly Report—same as above—they all log information. Depending on what your error is, you may pull different details from those reports. Then finally the Test Report—there are a few summary reports over in the C2C RI.

Ken Vaughn: The IEEE 829 Test Log is designed to provide a chronological record of relevant details of testing. It should generally log as much information as possible. The C2C RI divides the logs into several distinct reports, due to the size these reports can be. The reports include test case details, which is a step-by-step log of the individual steps defined in the test procedure; a message summary, which identifies each message sent or received by the application, along with a timestamp of that message; the message details, which shows the complete details of each message sent and received by the application; and then the script log, which provides a log entry every time the test enters or exits a test script. All of these logs can be useful in diagnosing problems and should be incorporated into the final test log. In addition, the IEEE 829 log report may include any logs that are kept separately from the C2C RI. For example, a separate line analyzer tool may provide its own log. Likewise, the tester himself or herself may keep their own written notes that should also be included in the formal IEEE 829 log report.

Ken Vaughn: An Anomaly Report is intended to document any event during the testing process that requires investigation. Any of the C2C RI log reports may be able to assist in identifying these anomalies. Rather than providing a complete log, the anomaly report should analyze that information and provide a concise description of the problem, using that log information as necessary. It should also identify the impact of the issue and any corrective action that may be needed. The C2C RI does not provide this level of analysis—only some useful tools to enable the tester to perform that analysis.

Ken Vaughn: Finally, the IEEE 829 defines a test report. This is a summary report that provides evaluation and recommendations for the entire system under test. It should include both the Test Case Summary from the C2C RI—which identifies the pass/fail status of each test as performed in chronological order—and the Conformance Compliance Report—which indicates the pass/fail status for each requirement, based on reversing the traceability tables. For example, if a test fails, the associated requirements will also be marked as failed. Now, Version 2 of the C2C RI will also allow the user to generate a Section 1201 Conformance Report. Section 1201 is the portion of the Safe Accountable Flexible Efficient Transportation Equity Act: A Legacy for Users, more commonly known as SAFETEA-LU. This is the legislation that ensures agencies offer real-time monitoring and sharing of travel conditions of major highways across the U.S. The Section 1201 Conformance Report will show the pass/fail status for each Section 1201 requirement defined in the legislation, and then associate that to the individual tests that were performed.

Ken Vaughn: That brings us to our next pop quiz.

Ken Vaughn: Which C2C RI report will assist in developing an Anomaly Report? Test case details; Message summary; Message details; or all of the above?

Ken Vaughn: The correct is all of the above. All of those reports can assist in developing an Anomaly Report, but they aren't the Anomaly Report themselves. The Test Case Details report will help identify the step where the error occurs; but other reports can assist as well, depending on the type of error. The Message Summary will help identify if messages were sent and received; but once again, the other reports help as well. And the Message Detail report will help identify if a message contained an error, make sure it's properly formatted, and all of that—but once again, other reports help as well.

Ken Vaughn: That completes our four modules—installing and configuring the C2C RI on a PC; operating the C2C RI; retrieving the C2C RI results from a test; and preparing a report based on those C2C RI results.

Ken Vaughn: The TMDD Testing curriculum is now completed. We started out with the module A321a, which discussed the user needs for the TMDD. A321b talked about the requirements associated with the TMDD. Module T321 talked about applying your test plan in general to the TMDD standard. Module T251

was an introduction to the tool that we just saw here—the Reference Implementation. Finally this course—T351—actually stepped you through the process of applying the C2C RI reference implementation.

Ken Vaughn: We thank you for your participation in this course and we ask for your feedback. You can use the feedback link below to provide us with any of your thoughts and comments about the value of the training. We do value your input and we try to constantly improve our courses. So thanks a lot for your time and we look forward to hearing from you. Thanks.