Vincent Valdes: ITS Standards can facilitate the deployment of interoperable ITS systems, and make it easier to develop and deploy regionally integrated transportation systems. Transit standards have been developed by transit professionals like you at a national level to encourage competition and limit costs within our industry. However, these benefits can only be realized if you know how to write them into your specifications and test them. There are now a series of modules for public transportation providers that cover practical applications for promoting multi-modalism and interoperability in acquiring and testing standards-based ITS Transit systems.

Scott Altman: Hello. This is Module 14, Part 1, Applying the General Transit Feed Specification (GTFS) to Your Agency.

Scott Altman: My name is Scott Altman and I have been a member of the technical staff at Consensus Systems Technologies, ConSysTec, since 2011. I have experience with various specifications for transit data, including transit schedule data and real-time information. I currently manage quality control and testing of GTFS data for the New York State Department of Transportation's 511 New York Transit Trip Planner. This effort involves collecting and creating GTFS, covering approximately 100 transit operators throughout the state. I have a bachelor's degree in civil engineering from Rensselaer Polytechnic Institute, and I'm registered as an intern engineer, or EIT, in New York State.

Scott Altman: Here are our learning objectives today, and they are, number one, define the scope of, uses for, and users of the GTFS specification; two, apply the steps for translating your transit source data to GTFS files; three, improve GTFS data quality; and four, illustrate how an agency implements GTFS.

Scott Altman: Our first learning objective today is to define the scope of, uses for, and users of the GTFS specification. The key points we will discuss are what is the background of GTFS, and the benefits and uses of GTFS.

Scott Altman: Before we begin learning about GTFS is, we will start with a story about a transit customer. Imagine you're traveling in a city you're unfamiliar with and you've landed at the airport and need to get to your hotel in the center of the city using only transit. Let's think about how you would do this. In the past you would have had to find a route map and a timetable. If your trip involved multiple agencies you may have had to track down multiple route maps and multiple timetables. However, today, we have a new options, and using our smartphones that we all carry in our pockets, we can use a trip planning app and plan our trip. The application can detect our location and we can give it our destination. Right now on the screen you see a screenshot of one of these apps where we've put in our origin and destination and we see a list of itinerary options. We can select the option that is the best, and we get a more detailed itinerary that tells us exactly what stops we need to start and stop at and where we're going to transfer. It also gives us our estimated time of arrival at our destination, and we can also get a map and see how to get from Point A to Point B using transit by looking at the map. So today we will learn essentially how this information gets from a transit agency to the customer's smartphone.

Scott Altman: So now we will begin by learning about what GTFS is. GTFS is an acronym for General Transit Feed Specification. GTFS is a commonly used format for disseminating static transit data. Another way to think about that is that it is a set of rules describing the format for sharing transit schedule information. What does this mean? This means that GTFS is not software, it's not a solution, it's not a system; it simply is a method we can use to get our transit information from our agency into an application that can use this information in a common format that is widely accepted. Now, throughout this module today we will use the term static schedule. It's important to know what this means, and in a transit context, a static schedule means one that is the baseline schedule. This is a fixed schedule that doesn't vary on a day-to-day basis. That's contrasted with real-time information, which can vary by the minute. In addition to providing the schedule, the GTFS format also describes other relevant data to transit trip planning, such as fare information, stop locations, and general information about the agency. One thing about the GTFS specification that is important to realize is that GTFS is not a standard. It doesn't go through a formal standardization process with a standards development organization such as ITE, APTA, or NEMA, and it does not go through a rigidly defined balloting process like formal standards might do. It is just a specification. However, it is often referred to as the de facto standard because it is widely accepted and extremely prominent and it is, essentially, the go-to way to share transit schedule data.

Scott Altman: Let's talk a little about the history of GTFS. In the late 1990s, the internet became a commonly used tool for planning car trips or to get driving directions. Instead of looking at a map to find where you were going, you could type in your origin and destination and you would get driving directions, and in 2005 Google Maps was launched, and that provided driving directions to the public. This led to a vision that transit trips could also be planned using Google Maps, and in 2005, the Google Transit Trip Planner was also created, with TriMet in Portland, Oregon, being the first agency. Originally GTFS was called the Google Transit Feed Specification, and was delivered as a partnership between Google and TriMet. The purpose was to get transit information into Google Maps, which is why Google was in the name of the specification. In 2006, the specification was formally released, and five cities began participating: Eugene, Oregon; Honolulu, Hawaii; Pittsburgh, Pennsylvania; Seattle, Washington; and Tampa, Florida. By 2010, hundreds of agencies were represented in the Google Maps. Early on in the process, Google supported agencies by helping them with the process of developing their feeds. Over time, this became unfeasible because of the large number of participating agencies. These days, the burden of developing GTFS feeds falls on transit agencies and their consultants. Today, the number of agencies that are providing data is in the thousands. Because of the wide acceptance of the GTFS specification, in 2010 the name of the specification was changed to General Transit Feed Specification, being the G stood for General instead of Google, and this is really symbolic of how GTFS is used by many applications, not just Google Maps.

Scott Altman: On the screen right now we see a map showing acceptance and implementation of GTFS throughout the world. We see the dark-colored countries and those are countries where GTFS exists somewhere in the country. However, there may be gaps within one of these countries. For example, the United States is shaded dark because there are many agencies that have GTFS, but there also are many agencies that don't currently have GTFS. We see a lot of countries where there is no GTFS at all, and this is big chunks of the world. Often in some of these countries it is very difficult to get the structure of transit schedules into a common format because the transit services there don't necessarily follow a fixed schedule. However, with modern technology, there are efforts underway to collect data that can summarize this data and eventually put this into a standardized format such as GTFS so that customers in those regions can benefit from transit planning-- transit trip planning.

Scott Altman: Now we will shift gears a little bit and talk about the process for updating the specification. The GTFS specification is officially maintained by Google. In this role, Google hosts the specification website, moderates the discussion group that discusses updates to the specification, publishes changes, and develops open source tools that can be used with GTFS. As we've discussed before, GTFS is not a standard and there is no formal standardization process. There is a structure to the process. It is slightly less formal than the balloting process used by more formal standards. With GTFS, discussion of updates occurs in an online discussion forum. This forum is hosted by Google. Membership is open to anyone who can claim some relationship to GTFS. For example, transit agency staff, application developers, planning tool users or enthusiasts have all joined and proposed changes. There is no fixed schedule where the specification is updated. In recent years it's been about once per year but it can be more frequent or less frequent depending on the needs of the users of the specification. Using the GTFS specification and the process for updating it offers flexibility. This can be seen as an advantage because changes can be made to the specification on an as-needed basis, and the barriers are relatively low to change relative to other standards. However, this flexibility may also be thought of as a disadvantage. Different classes of users of the specification have different needs, and there is not necessarily consistency with regards to which changes get implemented. Additional, irregular updates to the specification may mean that the timing of changes to software that either produce or consume or use GTFS may not occur on a consistent basis. However, changes to the specification are typically backwards-compatible, meaning that producers of the feed usually only have to make changes to their systems to incorporate new functionality, so older functionality that they have already been implementing should work the same.

Scott Altman: Now we will look at some details of what the process that is used to make changes, and the process begins with somebody in the discussion group making a proposal. Any member of the group may want to add a new field, for example. Once they make that proposed change, it is discussed in the discussion group and feedback is given. Sometimes this feedback may lead to a revision of the proposal. Once there is a general consensus to move on, a testing process occurs. For this testing process, one producer and one consumer must implement this change. That means a transit agency or someone who generates GTFS feeds must implement this proposed change, and then an application, such as a trip-planning application that uses GTFS, must also consume and use that changed field. Once testing is complete, there are more discussions and final comments are given. In the event that the users group members don't particularly like how the change was implemented, it goes back to the drawing board. However, if there is a general consensus, then the change is implemented and becomes part of the GTFS specification.

Scott Altman: Now we'll talk about some of the benefits of implementing the GTFS. If your agency was looking to implement GTFS, here are some advantages.

Scott Altman: One is that the specification is open. This means that anybody can access it and understand its contents. Additionally, anyone can use GTFS without paying any licensing fee. The full specification is available for anybody to read and there's no fee to download the specification or to look at the website where it's maintained. A second advantage is that the specification is flexible, meaning there are ways to accommodate most data requirements. In some cases, such as deriving a service calendar, there are several ways to do what you need to do. GTFS has also been widely adopted. In fact, it is the most commonly used specification for sharing static schedule data. Leading trip planners and other applications use this format. And the first four benefits that we talked about are reasons that you may select the GTFS specification over another specification. However, the fifth benefit is really a benefit to any specification where schedule data is shared. The final benefit is that GTFS enables schedule and route information to passengers. This may increase customer satisfaction because they have more access to information and ultimately could have benefits such as improved ridership if transit customers are extremely satisfied with the information that's provided to them.

Scott Altman: So now that we've learned about why you should use GTFS, we'll learn how it might be used. There are several classes of applications that use the GTFS specification. We'll talk first about some downstream applications. Downstream applications are tools that use GTFS, produced by agency as an input. These applications might be at the agency itself, or might be hosted by a third-party. This includes tools such as trip planners, which the public uses to select an origin and destination and get a response about their transit itinerary. Another group of applications that use the GTFS specifications are real-time information tools. Many agencies have applications that give up-to-the-minute information about transit service. Often a static GTFS feed is used as an input to these systems to give the baseline that the real-time information is relative to. Other operation systems at an agency, such as computer-aided dispatch or CAD systems, automatic vehicle location or AVL systems, or fare collection systems, might use a GTFS feed as a source data to help operate their system, and there are also customer-facing timetable generator tools where a GTFS feed can be taken and put into a filmmaker timetable format that the public might use. Some customers may prefer looking at a timetable over a trip planner. A second class of applications and uses of GTFS is in transit planning, and I don't mean transit trip planning tools, but I mean service planning and how-- determining what services an agency will provide. A GTFS could be input to a service coverage map, because GTFS has the data in it to define the transit network, the geographic network, and the service. This same day it can be used for service evaluation. An agency or planning staff could help drive service frequency. Additionally, the data that's used can be used to find gaps in transit service, so if there are geographic areas that may not have sufficient transit coverage, or it might be times of the day-- GTFS can be an input to data models that work for transit planning.

Scott Altman: We have an activity.

Scott Altman: And we have our first question. The question is: Which of the following choices best describes the process for updates to the GTFS specification? Our choices are A, formal balloting process by committee; B, discussion in an online forum; C, voting by every transit agency; or D, there is no process.

Scott Altman: And let's review our answers. The correct answer is B, discussion in an online forum. There's an online group where changes to the q specification are discussed before they are officially implemented by Google. And let's look at the incorrect choices. A is incorrect. There is no formal balloting process by committee because there is no formal committee that exists. C is incorrect because every transit agency does not have to vote. While there is a discussion process, there's no requirement to vote, and not everybody participates. And D, there is no process. This is incorrect, because even though the process is informal, it does exist. If there was no process, changes may never occur.

Scott Altman: And here is our second learning objective that we will discuss today: Apply the steps for translating your transit source data to GTFS files. Our key points for the second learning objective are inputs needed for GTFS, GTFS structures and files, translating source data to GTFS, special cases in GTFS, and using file parameters.

Scott Altman: So now we will begin learning about what makes up a GTFS feed and we will do that by looking at this diagram that shows different groups of information that must be collected. So if we start at the left of the diagram, we have to collect basic agency information and contact info. This is used for customers to know who to call if there is a problem or if they need more information. Moving in a clockwise manner, we need route information, as this is the basic service that many customers are familiar with. Customers typically refer to transit service by the route they take. We also need detailed schedules that provide the actual times so that this can be given back to customers. We might need pattern information. Patterns in the transit world are typically the sequence of stops and the times applied to those stops in generic fashion. So if there's a sequence of stops that's repeated throughout the day, that is referred to as a pattern. We need information about stops. This is basically so that transit customers know where to go. We need to know service calendar information so that we know whether or not a service is running on a specific day. Different agencies have different calendars for weekdays, weekends and holidays often. We need to know trip path data. That's the actual route that a transit vehicle takes, and that often shows-- reflects how a trip appears on an application using the feed. You can see a map and how the bus or train actually travels. We need fare information, and this is important so that customers know how to pay. Although this is still an actionable part of the specification, customers typically want to make fiscally responsible choices and might want to take the most cost-effective route.

Scott Altman: So here we'll discuss where the data comes from that is loaded into the GTFS feed, but before we do that, we should discuss the characteristics of all data, in that all data needs to be accurate, reliable, and consistent. This is important for ensuring that information is correct and usable by anyone who needs to use it. Today we live in a digital world. As such, many agencies have invested in electronic systems. In some cases, all of the data might already be collected by the agency. For example, many agencies have scheduling systems that they use to help generate their schedules, and often GTFS can be exported as an output from these systems. Agencies also often have CAD/AVL or computer-aided dispatch and automatic vehicle location systems. These systems often have to collect data about the transit networks and might have some of the data you need. There are all paper sources of data for agencies who have not yet invested in electronic systems. These include timetables that may be published for the public to see, and list the stops that may be contained. And finally, a very important resource in generating GTFS feeds is the staff of the agency. Scheduling, planning and operations staff all have a familiarity with the service and these staff members might be able to help provide what you need to create a GTFS feed. Additionally, agencies may employ data collection staff to collect data such as exact stop locations or path data that trips may take.

Scott Altman: Knowing the types of data we need, we will now talk about the structure. GTFS feed is a series of zipped text files in a CSV format. For example, there might be a file called FictionalGTFS_Spring2016.zip. What does what I said just mean? CSV means comma-separated values. As we see, we have an example of what one of these feeds might look like on the bottom of the screen, and we see there's a comma separating every field. Each of these feeds also has a header row. This describes what each field is. So for example, we might have a field for route identifier or a route name in the header line, and then the first field would be a route identifier and the second would be a route name, and we would have that moving down. So each row of information is a new data point or a data frame, is another term you may be familiar with. And note the GTFS files, files within a feed, are saved as .txt or .text files. Even though they're in the format of a .csv file, or a comma-separated value file, they are text files and you can open them in a commonly used text editor, and we will see what that means in a minute.

Scott Altman: So here we see an example of a GTFS file viewed two different ways. The first way we're going to look at is a text file. This is the route.txt file, and we will discuss what that means in a minute. We see that this is a comma-separated value file, so we see we have fields, and there are commas separating each field. Now, this file you see here, this is a fictional example, and we see it's got a header row and then rows of data. One thing I'd like to point out is that in the second and fourth line we see quotes around the second line, "Bus service to downtown area of Anytown, USA." The reason we have that quote in there is because we see there's a comma in there between Anytown and USA. If we didn't have those quotes, because this is a comma-separated value file, an application reading this would think that that's two different fields, but because it's not two different fields, it's one field, the quotes tell the application reading the file to ignore the common within the quotes. A second way of looking at this type of file is in a spreadsheet editor such as Microsoft Excel, which you may be familiar with. We see here we have the same header row and we have subsequent rows of data. You'll notice that the quotes don't appear because when it was read into Excel, it knew to ignore the comma, and it doesn't actually show the quotes, but they are there in the source. If we were to export this, then the quotes would show up again in the text version. And another thing to look at is that we see the comparison. The top row is the header on both files, and we see how a column of data in the text format translates to a possibly more familiar comma in a spreadsheet format.

Scott Altman: Here we'll talk a little more about the structure of GTFS. Since GTFS is not a standard, it differs in some ways from other ITS standards that you may be familiar with or may have learned about in other modules. A key difference is that many standards are structured into messages within a single file. Instead a GTFS feed is a series of files. So it's multiple text files in one zipped up feed, Zip feed or zip file, the terms are interchangeable. I generally will use for the rest of this module the term feed to discuss the set of all the files, the file to look at one set of data, one of the files. GTFS does not follow the typical request-response pattern that you may be familiar with from another ITS standard such as TCIP. Instead the GTFS feed is simply downloaded from a fixed point. We see a graphic that shows kind of a comparison of terms you may be familiar with from other standards. As I said, there's no dialog with GTFS so we don't see an equivalent. However, we see the message, which in another standard might be one file, and here it's the feed of zipped up files. Another standard, they may have a data frame, which is a collection of fields, and here we have a whole file, the rows file, and those are collections of fields. And finally, in other standards we use the term element to refer to the basic level of information. Here we'll use the term field. And this should make more sense as we look at some details of the actual specification.

Scott Altman: Here you see a list of the files that are contained within a GTFS feed. You can think of these are various data frames, if you will, that occur in GTFS. Of these files, the first five are mandatory. Every GTFS feed must have all five of those files with data included in them. The next two, calendar and calendar_dates.txt, you always must have at least one of them. However, they are interchangeable. According to the specification, the calendar.txt file, that is the one that is officially required. However, it can be substituted for calendar_dates.txt. Finally, we a bunch of files at the bottom, and those are optional files. If you don't have the data to populate those, you don't have to include those, but they do provide value to your downstream users, and as we go through, we will learn about the intent of each of these files.

Scott Altman: Here we will have a brief overview of the types of data that we may encounter as we go through the GTFS feed. These are the most common types of data. One common one is a String. That's a series of letters, numbers and characters. So we see an example string is "Grand Central Terminal." We also have numeric values. These are numbers that may or may not have a decimal point after them. And then we have integers, which are numbers that do not have a decimal point after them-- so 1, 2, 3. The concept of dates in GTFS-- dates always appear in the format YYYYMMDD, or year, year, year, year, month, month, day, day. So April 25 of 2016 would be 20160425. Time in GTFS always follows the 24-hour clock. So we see on the screen 23:10, and we always have hours, minutes, seconds. So 23:10:00. This may also be referred to as 23 hundred hours. That will, of course, translate to 11 o'clock p.m., if I say 23 hundred hours. The clock always starts in GTFS at zero, or zero hundred hours, and goes to 24 hundred hours, being midnight on the second day. If service starts on one day and extends beyond that day, it's okay for this number to extent past 24. So if a trip starts at 11:50 p.m., that would be 23:50:00, and if it ends at 1 o'clock p.m., we would refer to that as 25 hundred hours. Another data type that we will see are enumerated or restricted values. These might be integer values that translate to a specific meaning, or they might be a code that represents something, and we will discuss that as we go through. URLs are typical web links. And the final field type that we might see are hexadecimal colors. This is a series of six characters, either letters or numbers, that refer to a standardized list of colors, and we'll also see how that is used as we move forward.

Scott Altman: Now we will begin going through the actual details of what is contained in the GTFS specification, and we will look at each file. As we do that, the important thing to keep track of is the message or the intent of what each file represents and what type of information it transmits. It's not necessary to understand every detail, because that can be looked up in the GTFS specification reference, but what's important is to understand the concepts. Now, as we go through we will see that every file has optional fields and required fields. A field that is optional means that it is optional if that file is being implemented. So here we see there are some optional fields, but the file is required because this is a mandatory file. However, you don't have to include those optional fields, and typically there is no metadata to know what fields and are not present. We just must look at the file and know which fields are and are not included. So this first file that we have here, this is the agency.txt file. This represents basically agency correct information. So this describes the name of the agency, ways to contact the agency, including phone number, email address or web link. It also describes geographic information, very basic, such as the time zone, and demographic information such as the primary language that is spoken in the region. The agency_ID field exists. That's a unique identifier that you can assign. It's optional, but if you have multiple agencies within a single GTFS feed, which is common in regions with many different agencies operating, then you would need to use that to differentiate what service belongs to what agency. However, if your agency is just your agency, you don't have to have a unique identifier.

Scott Altman: Here's an example from the real world of what the agency.txt file looks like. A few things that you should note is we have a time zone in the agency time zone field-- America/New_York. That's a standardized code that corresponds to the Eastern time zone. We also see the last value on the line is .en. That's a standardized code for English, meaning that English is the primary language spoken in this agency's region. We also see that this agency has a phone number and web link, and you can see their agency name. And as you can see, the order of the fields here is different than they are on the specification. That's okay, as long as the required fields are present and the optional fields make sense. It doesn't matter what order you put them in.

Scott Altman: A second file that is part of GTFS specification is stops.txt. This is a mandatory file, and it essentially is a directory of stops. So when you have a trip and you're referencing what stop it stops at, you reference the unique identifier or the stop_ID contained in this field. Stop ID is typically an internally used code. It can be known to the public; it doesn't have to be known to the public. However, it is required that you have a unique identifier unique within your feed. You can also have an optional stop code. Some agencies have a service where you can send a text message to the agency and real-time information over the next bus or train arriving at that stop. A stop code is often a value that's published that you can use for that purpose. You also always must have a stop name so that the customer knows what stop you're talking about, and you also must specify a location using latitude and longitude. That is required in all cases.

Scott Altman: Here are some values that are contained in the stops.txt file. A few I'd like to point out the zone_ID is used for fare collection. So you can assign a fare zone if your agency uses that format. We can also define a stop as a stop within a parent station. So for example, you may have a large train station that has multiple tracks. You could differentiate those at different stops. If they're consistent for your trips and predictable, then you can define that within a parent station.

Scott Altman: Here's an example of what the stops.txt file looks like. We see the stops in all cases have a stop name, they have a latitude and longitude and a unique identifier. We see that some of the values are blank in some of the fields, and that's okay. Notice that the stop description field is used in the first entry only and not used in the others. That's okay. As long as they are represented and we see there's a blank space for that, that's okay.

Scott Altman: One field I'd like to point out, is the routes.txt file, and this is an important file because it describes a description of the routes in the system. Whether an agency operates one route or one hundred routes, this is the entity in the transit system that customers may be familiar with. The route_id is the unique identifier, and this may be a publicly known number or value, or it may not be, and that's okay, because we have the route_short_name and the route_long_name values, and those are the publicly known numbers, or route names. The route_short_name is typically a number or abbreviation, and the route_long_name is a more detailed description of the route, or name of the route. Notice that both fields are required. However, if you don't have the data for both, you can leave one blank, but it must be included. Another field I'd like to point your attention to is the route_type field. This describes the various modes that a route might take. For example, it might be a streetcar, a subway, a rail, a bus line, or etcetera. So we use the integer values that you see on the right of your screen, and those designate what it is, or what mode it is. We also should look at the route_color and route_text_color fields. Those are the color as well as the text color of a logo of the route typically. So if you have a route logo with a red color and white writing. That is used here, and those are hexadecimal values as we talked about before. We'll see that in an example in a moment.

Scott Altman: Here we see the routes.txt file, and notice that the route_id field is a seemingly random number-- 12577. However, we have a route_short_name, Sy 10, that is more familiar to customers. We also have a route_long_name, which here describes the starting and ending points of the route, or the origin and destination. You should also look at the route_color and route_text_color fields. We see in the second field there's a route color 7E3092. That's the code for purple, and we see the purple box. We also see the text color is FFFFFF. That's the code for white, and we see how the writing Sy 16 is white, and that also corresponds to the route short name in that line.

Scott Altman: Now that we've discussed the route structure, we'll talk about the actually trips.txt file, and this describes the trips that occur on a route. We need to link a trip to several things. It has to be linked to a route identifier so we know what route it's operating on, and a service ID, which is the calendar, and we'll talk about that in a moment. You may have a trip_short_name. That's a publicly known number for a trip. And we also could have a headsign. So if there's a headsign in front of a bus, we might designate this here so that a passenger would know what bus to get on. We also have a concept of direction ID. This is not a required field. However, trip planning tools often rely on this to show movement from one direction to another. Possible values are zero or one. It's up to an agency to determine what a zero means-- direction zero is-- what direction one is. It's not standardized, but it needs to be one of those two values. And we can also link to a shape, which is the actual path the vehicle takes, and we'll talk about the meaning of that a little later.

Scott Altman: Here's an example of a trips.txt file. We see it has a route-- it has a service_id counter of 58, a long, random trip ID, and then those headsign values. The first two trips have one headsign and the route direction of those is 1, and the second two trips have a different headsign and the route direction ID is zero, and that means the first two trips are going in one direction, the second two trips are going in the other direction, because we see they're all on the same route. And here's a graphic showing you what a headsign is in case you aren't familiar.

Scott Altman: Having discussed the trips, we'll now talk about how we determine the times and stops and sequence of each trip, and that's done in the stop_times.txt file. Before we talk about these, I'd like to discuss the concept of a time point. Many agencies use time points or control points to show the schedule times of major stops. These are the points that a transit trip strives to adhere to. We see a diagram, and the squares represent those time points. Often, however, a transit vehicle might make other stops between those time points, and those are represented by intermediate stops. Sometimes we don't have times for those intermediate stops-- agencies don't always publish that-- and that's okay. They can be left blank in the GTFS specification. However, it is the best practice to include those times when they are available so that customers know when they should be prepared to board a bus, train, or other mode of transit. So let's look at some of the important fields that are contained in the stop_times.txt file. We need to tie each trip to the trip identifier that we defined in trips.txt. We also need to have an arrival and departure time. Now, if we don't have those, they can be left blank. However, the field must still be present, and you can have a blank field, but it has to be there. Now, the first and last stop of every trip always must have an arrival and departure, and also, if we include an arrival time, we always have to have a departure time, even if they're the same. If we're including one, we have to include both in a line. You can't leave one blank or the other. We also have to link to a stop_ID as well as the sequence, so we know what order the trips-- the stops that occur on a trip.

Scott Altman: There's also some other fields that can help us distinguish whether we have a time point. The shape_dist_traveled, we can determine how long along the shape the trip has traveled, and we'll discuss that concept a little later, and we can discuss whether this is a regularly scheduled stop or it must be prearranged or coordinated.

Scott Altman: Here's an example of a small part of a stop_times.txt file. We see we have a trip and the first stop occurs at six o'clock. The second stop occurs at 6:05. The third stop occurs at 6:10. So the stop_times.txt file would have this whole trip and then the next whole trip, and they would be repeated, and it becomes a very large file.

Scott Altman: Here's the same trip that we've shown a part transposed onto a map so we can see how those times are translated. We have a time of six o'clock, a stop at 6:05, and we see a clip from the stop_times.txt file showing how it translates. So we see the six o'clock stop translates onto the map.

Scott Altman: So now that we've described the trips and the patterns that the trips take, now we will talk about how we determine service in terms of the calendar in GTFS. Calendars are a topic where there's really no one correct way to do it. Every agency has different operational services on different days, and there are different ways that you can designate a calendar. The one consistent thing is that every calendar must have a service identifier or service_ID. The calendar.txt file is the one where we can describe a recurring service-- so a service that occurs every Monday, every weekday, every Saturday or Sunday or every weekend, and we do that by having a series of fields for each day of the week and we can say zero if the calendar doesn't operate on that day, or one if that one does operate on that day. Each calendar can also have a starting and ending date, and I'll explain this in a little more detail on this example. So we see three service calendars for this agency. The first calendar has a service identifier of week, and we see that that one's a weekday service. It operates Monday, Tuesday, Wednesday, Thursday, Friday, as we see ones in each of those fields. We see zeroes in the Saturday and Sunday fields because this service does not operate on weekends. So a trip that has a service_ID of week only operates on those days. Below that we see Saturday and Sunday service. The Saturday service only has service on Saturday, designated by a one, in the Saturday column.

Scott Altman: There's another file that we can use to also help us define a service calendar. This is the calendar_dates.txt file. The primary intent of this file is to describe exceptions to the calendar.txt file. So we had a service calendar with service_ID. If that service were to not operate on a day of the week-- for example, the weekday schedule may not operate on a holiday-- then we would see that date and we would put an exception type of two, because we can see that that means that that service is removed or does not operate. If a different service were to operate on that day, we would list that service ID and that date and an exception type of one, and that means a service is added. We call that a type one exception. Another way of using this file is to list all of the dates that a service operates with a type one exception. So you could have some agencies that have a big calendar_dates file that lists every day the service operates. They don't define service as Monday, Tuesday, Wednesday, Thursday, & Friday. They might describe dates. That's acceptable in the GTFS specification.

Scott Altman: This example should explain these two concepts. These two examples. The first example we see shows you the calendar_dates file as an exception. So we see that on 20160530, which happened to be May 30 of 2016, which happened to be Memorial Day, weekday service had a type two exception, meaning there was the normal weekday service, and trips that have that service ID did not operate that day. Instead, Sunday service was operated, which might be more limited and more appropriate for holiday. We also see the second type of use of this file, and here we have a special service ID, and that ID has service the five days we have listed-- July 1st, 2nd, 3rd, 4th and 5th. So those all have a type one exception, meaning that a trip ID with that service ID of "special" operates on just those days.

Scott Altman: Now we move into the portion of the specification that is optional, and so we will describe a bunch of files that add value if you can provide them, but if you can't provide them, it doesn't mean that you shouldn't produce GTFS. So the concept of fares is not necessarily a complete description of fares for an agency. Fare information has been described in the GTFS specification for a while, but different agencies have different ways of disseminating fares and there is not a one-size-fits-all way of viewing this. So the fare_attributes and fare_rules file, which we'll address in a minute, describe the way that it's currently supported in the GTFS specification of describing your fares. However, there are working groups that are looking at other ways of incorporating fare information to accommodate different fare patterns that are used throughout different agencies. So the fare_attributes file describes an instance of a fare. So it describes a fare identifier with price and the currency. It can also describe whether you pay the fare onboard or if transfers are included. Note that there currently is no way to determine different classes of fares. So if you have a different fare for senior citizens or a different fare for students, there's no way to differentiate that between the standard fare. There's also no way to differentiate a peak fare from an off-peak fare, and no way to differentiate a different type of fare media. For example, an agency might charge a different cash fare versus a fare that's bought on their fare card. So it can really only describe the main instance of a fare. That's an important point to note if you wish to describe fare information.

Scott Altman: Here's an example of the fare_attributes.txt file. We see this agency has a flat fare for trips that have a fare ID of base. They have a different flat fare for a different route of that fare ID of baseBP. And then they have different zone-based fares, and we'll discuss how we know that in a minute when we discuss the next slide. So just keep this example in mind as we move on.

Scott Altman: The fare_rules.txt files describes when each fare is implemented. So we can link it to the fare_attributes using fare ID and we can describe a route_ID for the fare. So if we just include a fare_ID and a route_ID,that would mean that all fares on that route have that fare-- all trips on that route have that fare. However, there are also ways to describe-- we can describe a fare based on the origin_ID, the destination_ID, with contains ID, which describes if a fare or a trip passes through that ID, and by ID I mean that zone, and stops we can define-- in the stops.txt we were able to define the zone, and this file is where we would use that zone. So an origin ID is a zone, a destination ID is a fare zone, and the example that I will show in a moment should explain that a little more clearly.

Scott Altman: Here we see a clip of the fare_rules.txt file from one agency. So the first fare is what they call their base fare. We see one route listed, but if there are other routes that use that fare-- and in fact, in this agency there are-- we would list that base fare with all of the routes that use that fare. On one route we see they have a different fare, and we list that-- that's the baseBP fare. Then on another route, which is ID 540-155, we see they have a zone-based fare. So the first fare ID, that's implemented on all trips originating from the zone with the origin ID of 1. A second one is for all trips originating from zone ID 2. Some agencies might also pair a destination ID-- so we can say all trips that originate from origin ID 1 and end at destination ID 20 have this fare. There are different use cases for how we would view this. So this is the only concept that, as I said, is still being worked on and improved.

Scott Altman: Another optional file that we have in the GTFS specification is the shapes.txt file. This describes the exact path that a transit vehicle travels. This is an optional field. However, it can be included and it should be included when available so that when a trip is drawn on a map, the shape looks correct and it makes it more appealing, and we define shapes by a series of latitude and longitude points, as well as putting them into a sequence, and we also contract difference traveled. This is sometimes used for complex shapes to match that same value in the stop_times.txt file to the shapes file, and it helps trip planning tools to more clearly draw the shape.

Scott Altman: Here's an example of what the file looks like, and the data that we have that populates this might be collected from another system the agency has, such as an AVL system, or there are software tools that can help you determine the shape.

Scott Altman: And here's what a shape looks like. We see a shape that's transposed on a satellite view. This is one whole trip shape that we see here.

Scott Altman: Another way of describing trips is the headway, frequency-based trip. Some agencies don't operate their trips on a fixed schedule. They might operate their services on a headway, such as every five minutes, every ten minutes, and they don't define exact times. So in such a case, we will still define the trips in the trips.txt file. We would still find them in the stop_times.txt file. Instead of we would describe times of the trip, we would really just be describing the travel time between stops, and in the frequencies file we describe the start and end time of a specific headway and that headway in seconds.

Scott Altman: So in this example here, we see we have the same trip ID through four periods of the day. From six o'clock a.m. to nine o'clock a.m., this trip operates with a 600-second or 10-minute headway. From 9:01 a.m. to 16:00 hours, or 4 o'clock p.m., this trip operates with a 1200-second or 20-minute headway. So we can define trips that way instead of having to hold ourselves to specific times.

Scott Altman: Now, often when a customer is planning a transit trip, they need to transfer from one route to another and may need to walk from one stop to another to do that. If there's an official transfer, you can include that into the transfers.txt file. In such a case, you could define a from_stop_id, or the stop that the vehicle first arrives at. Then the customer would get off and have to walk to the to_stop_id, which is where they will depart on their next trip on a different route typically.

Scott Altman: So here's an example, and this shows a trip that starts at a transfer. So the customer would arrive at stop 1210. They would walk to stop 1490, and the minimum time required to do that is 120 seconds, or 2 minutes.

Scott Altman: So we come here to the final file in the GTFS specification, and this is the feed_info.txt file. This describes metadata about the feed. So this doesn't describe customer-facing information, but it describes information that describes the feed itself. For example, if the feed is published by someone-- it might be the agency or someone other than the agency-- we could attribute them here. We can also describe a start and end date of the feed and the version of the feed.

Scott Altman: Here's a sample of what that feed looks like, and we see here the actual agency published that feed. So now we've learned about the GTFS specification and all of the files and fields contained within it.

Scott Altman: Now we will talk about the tools available for creating GTFS and editing GTFS. There are several ways an agency can create GTFS files. An agency could go in and manually write each file or manually create each file. This does take time, but it is manageable for small agencies that may have a limited number of routes, trips and stops. Some agencies might have their data in a format different than the GTFS format. In that case, it may be beneficial or useful to create custom software that can translate from one format, a native format, into GTFS. Other agencies may have scheduling systems, and they can simply export GTFS from scheduling systems. The feature may be included in the scheduling system; they may have to pay for an add-on to the system. Finally, there are off-the-shelf tools that can be used. Some are open source tools that anybody can obtain, and some are very accessible tools, such as the National RTAP GTFS Builder, and we will discuss that over the next slide.

Scott Altman: So the National RTAP GTFS Builder is a tool produced by the National Rural Transit Assistance Program, or RTAP. This program was created in 1987 and it is a federally funded program of the Federal Transit Administration that provides support to rural transit agencies, and to help agencies create GTFS, they created a product, as I said, known as GTFS Builder. This is free, and it's based on and even relies on Microsoft Excel, which many agency staff are likely to be familiar with. This tool is an easy way for simple transit systems to create their data in GTFS. It works best for agencies with a small number of routes. In order to use GTFS Builder, it is still critical to have all the data that we described, and then this tool has template spreadsheets that we can use to build a schedule, and it has macros, or custom software, build into it to translate those template spreadsheets into GTFS files. So the information we talked about with stop locations, calendar information, patterns and time information, those are all inputs to the tool. At the top of the screen we see the link that you can use. As you sign up for a free account with National RTAP, you can access this tool, and there are instructional videos that are available. And we see on this screen the familiar Excel format of the tool, and you can see this graphic a little more clearly in the Student Supplement.

Scott Altman: Here we describe some special cases that may occur in the GTFS specification, and these are just some of many scenarios you may encounter. So the first scenario says, "My service calendars don't follow traditional schedules, such as weekday, weekend, Sunday, Saturday, etcetera." In this case, if they don't follow traditional schedules, we talked about how there's an alternative way of building schedules in the calendar_dates file by just listing all of the dates out for that calendar and link to a type one exception, so we can do that. Another special case is if you don't have exact times for intermediate stops between time points. In such a case you can omit those time points. Downstream applications typically have a mechanism to interpolate that data. Otherwise, it just might be blank and if you don't have data you're just simply not providing customers with incorrect data. A final scenario is, "I have flex-route service. How do I account for this?" There's currently a working group exploring this, but there is no consensus at this time on how to accommodate flex-route service. But in the future this might be something addressed by the specification.

Scott Altman: You may also come across a scenario where the GTFS doesn't describe something that you need to describe in your feed. GTFS feeds can be extended. You can add a whole extra file, or you can add extra fields to the existing files. Extending a GTFS feed does not mean that that everyone will be able to use the files or feeds. It's not part of the specification. There's no guarantee that they will be universally accepted. However, you can still do this, and it's a good thing to do because in order to change the GTFS specification, you have to show that you've implemented-- if you propose a change, as I said, you need to show that this is being implemented somewhere and you could show an extension with your change as part of the testing step of the specification process. So, hypothetical examples, you might want to add a field to show whether there's a shelter at a stop. So you could add a field "shelter" to stops.txt. You may also want to add a file to describe parking facilities or parking.txt.

Scott Altman: And those are just some examples of many exceptions. So we come to our next activity of the day.

Scott Altman: And our next question is: Which of the following areas is not described by the GTFS specification? Your choices are A, stop locations; B, transit trip stop times; C, fare information; or D, historical ridership data.

Scott Altman: And let's look at our answers. Our correct answer is D. Historical ridership data is not included in the GTFS specification. If an agency had a need to do this, they could extend their GTFS. Other answers: A is incorrect because stop locations are covered in stops.txt. B is incorrect because transit trip stop times are covered in stop_times.txt. And C is incorrect be fare information is covered in fare_attributes and fare_rules.txt.

Scott Altman: We have a second question for this learning objective, and that question is: Which of the following is not a tool that can be used to produce GTFS feeds? Your choices are A, pencil and paper; B, scheduling software add-ons; C, the National RTAP GTFS Editor; or D, open-source software.

Scott Altman: And the correct answer is A, pencil and paper. This is not possible to use this because it does not create electronic files. The other choices: B is incorrect because many scheduling software systems do provide the capability to have add-ons. C is not correct because the National RTAP GTFS Builder can be used to create GTFS. And D is incorrect because there are open-source tools, as I mentioned, that can be used to create GTFS feeds.

Scott Altman: So as we move along, we move on to our third learning objective, Improve GTFS Data Quality. The key points here that we will discuss are testing GTFS files, describing data quality through metadata and versioning of GTFS files, using use cases to understand data, and improving data quality through data management practices.

Scott Altman: So we will talk for a few slides about the importance of testing GTFS files. Testing GTFS is very important because it ensures that the feed is usable. When I say ensuring that the feed is usable, that has two meanings. The first is to ensure that the feed meets the requirements of the GTFS specification, and the second is to ensure that the feed accurately reflects service and is meaningful to individuals who will ultimately use the data. GTFS that fails either of these layers of testing are not necessarily usable or are generally not usable by downstream applications. As I said, the first half of testing involves verifying the GTFS conforms to the GTFS specification. So here we would perform tests such as ensuring that all mandatory fields and files are present, all data is in the appropriate form, all references to other files make sense, as well as checking-- so we might check the trip_id in stop_times.txt also exist in trips.txt, and we also want to check that nothing impossible is happening. For example, we want to make sure that a trip does not leave a stop before it arrives at the stop. That would be in conformant to the specification. The other thing we check requires more human interaction to make sure that data is accurate. Someone familiar with the agency, typically an agency staff member, should verify that all the details are correct, such as stop location, stop times, trip times are correct, in order to make sure-- and other things like fares are correct, and agency contact information. No amount of automation can fully perform this part of testing, although it can be used to provide warnings that may lead to the discovery of problems. Tools that allow the tester to see data in a more user-friendly form, such as a timetable, map, or emulation of a live trip planner, can be helpful for this, and we will see an example of that in a minute.

Scott Altman: But first we'll look at a tool that is used to help us check that our feed conforms to the specification. The GTFS Feed Validator Tool is a commonly used open-source tool that is published by Google, the maintainer of the GTFS specification. This tool verifies conformance of the feed, and it takes a matter of minutes to test that a feed conforms to the specification. This tool also provides errors when it finds a problem, and warnings that might indicate some other accuracy error but may not be an error, and we will discuss what that means. But first here we see an example of an error, and the wording of these is a little bit arcane, but basically what this issue is saying is that there's an invalid field in the departure time, and what that means is the departure time occurs at an earlier time than the arrival time. So the bus is departing a stop before it arrives at that stop, and that is not physically possible if you consider the laws of time. So that produces an error, and the same error I mentioned on top is what we see an example of.

Scott Altman: This tool also provides warnings. Earlier I talked about how a route_short_name should not contain a route_long_name. They should be different. And here's a warning just highlighting-- or an example of a warning highlighting that. Or it mentions a warning, an example of that warning. However, another common warning that we see in the image is high-speed travel detected. This means that a trip travels between two points in a faster-than-expected time. Sometimes this is not a problem. You just need to verify that and go into your data and look and see if this is actually an issue or not when it occurs.

Scott Altman: Now that we've discussed tools for conformance, there's also a tool that you can use to help verify the accuracy of data. This is the GTFS Schedule Viewer Tool. This is also an open-source tool and is also provided by Google as part of their suite of tools for GTFS. So here we see this is a user-friendly interpretation of a trip. We see a big map in the middle with a trip and a time for every stop shown. So we could use this to verify that the location of the stop is correct and the time is correct, and we also see other information on the bottom, such as the route identifier. And we can select the list of routes on the side; we can search for specific trip identifiers or times.

Scott Altman: While being loaded with new GTFS, many tools will often overwrite current data. For example, if GTFS feeds contain a schedule information starting June 1, when you load it to the downstream application on May 20, that application might overwrite any previous data. To avoid this problem, we merge feeds together-- we merge a current feed and a future feed together so that way when we load it to application there is no gap in data. Google provides a tool for this called the GTFS Merge Tool, and I'm going to highlight what that means. So we have an agency here that publishes four schedules a year. They have a winter, a spring, a summer and a fall schedule. They released a GTFS with their spring schedule before the end of winter so that downstream applications have time to incorporate it. So when they do that, they need to merge the winter feed and the spring feed so that when they load it to the trip planner, there is no gap in service. So this is a common thing that's done. Google requires that feeds that are submitted contain current-- if you're submitting for a future change that you merge the future change with the current service.

Scott Altman: Another example of GTFS merging, we talk about the feed-- in the metadata file that we discussed, there was a feed_start_date and a feed_end_date. So we could use this file as another step in quality control to make sure that all the data we have-- that we have all the data we need. So we discussed that the feed_start_date and the feed_end_date might not have to line up with the dates in the calendar, but this could lead to gaps in coverage in your feed. A feed_start_date, for example, might be earlier than when there is service defined by the calendar. So here we see an example of a feed_start_date that starts on September 1, but a calendar that runs from September 15 to December 30. So this means that this feed has no data from September 1 until September 15. So looking at this we see a potential problem there. Additionally, this feed has a feed_end_date of December 15, even though the calendar extends to December 30. This should be interpreted to mean that after December 15, data may not be reliable and new feeds should be expected. Another aspect of the feed_info.txt which provides metadata we can use is the version field. This helps keep track of the version of the feed and might be usable to find a problem or refer-- might be used for archiving purposes or it might be used to send users of the feed back to another version.

Scott Altman: So we've been discussing the importance of providing high-quality GTFS data. So data that conforms to the GTFS specification must be accurate. However, it is equally important to ensure that the correct data is being provided in the GTFS feed. There is no use in providing the best data if it will not be used by downstream users. Therefore, high-quality data must also be useful data. Therefore, when determining what data to include, we might want to prioritize the data that is included. This is done by knowing your downstream users and how they will use the data. Few GTFS feeds contain all 13 GTFS files, and so if we know what are users will be using, then we may know what to include. For example, if our downstream users are a trip-planning application, then we need to include all of the mandatory files. We also might want to include the shapes.txt file, which describes how the shape will be drawn on a map. If there's a fare calculator, we might want to use the fare_attributes and fare_rules files. If you have customers who bike to transit, you might want to incorporate the bicycles_allowed file. Customers who require wheelchair accessibility-- if you have customers who do that, you might include the wheelchairs_boarding field, and this is important particularly with Americans with Disabilities Act, or ADA compliance. You want to know how customers who require such services can access them. So in understanding data, a use case is sometimes helpful. So we see here the use case of a customer who has a mobile device and wishes to get data through a trip planner. The customer requests an itinerary from the trip planner, the trip planner provides a response with the itinerary. Additionally, the itinerary is fed by GTFS feeds from an agency.

Scott Altman: In talking about providing high-quality data, we finally want to talk about the best practices for data. Google provides a list of best practices. Best practices are recommendations to organizations producing GTFS, which when implemented lead to highly usable GTFS feeds. These are not requirements but they are useful to ensure that your data is as good as possible. There are three main components to best practices. The first best practice is to provide high-quality data. Some of this is stuff that improves the appearance of the data. For example, the route_color and route_text_color, they should be contrasting so they're readable if they are shown on a map. You might want to use parent stations containing the child stops to help describe large transit facilities that have smaller stops between them. You want to ensure you're showing accurate shapes so that the maps look as good as possible. And you want to make sure that the route_long_name does not contain the route_short_name. Often tools that show the GTFS data will show the route_short_name right next to the route_long_name. So if the route_short_name is a route number and the route_long_name contains that route number, you might see the route_short_name appear twice, and it just doesn't look good. But again, these are all recommendations, not requirements. You also should test data when possible, and as I said, you want to test that the data meets the specification and that it properly represents service. Failure to test data may lead to incorrect dissemination of data. A passenger might go to the wrong stop location if they're not told the correct stop, or they might miss a transit vehicle if the time is incorrect. And finally, you want to make sure you keep the feeds up to date. If agencies schedule changes and the GTFS changes, this is so you can ensure that incorrect data is not being provided to customers. Incorrect information may cause customers to miss a bus or train, whereas it might be better to not provide information if you don't have it, because then a customer may just go to an alternate source and get the correct data. So sometimes no data is better than wrong data. But when you're providing data, it's good to keep it up to date. A good rule of thumb is to provide a new feed to downstream users two weeks ahead of time. This has multiple purposes-- one, to ensure that it gets incorporated into their applications properly, and it also allows time to find any problems and get them rectified before that schedule takes effect. Google provides a list of best practices and you can access it by going to this website.

Scott Altman: We have our next activity.

Scott Altman: And our question here is: Why is it important to test GTFS feeds? Our choices are, A, ensure that data is accurate and conforms to the specification; B, ensure customers use trip planners; C, so agencies can change schedules on the fly; and D, because testing requires no effort.

Scott Altman: Our choices are here, and our correct answer is A. Without testing data, it is not possible to know whether it can be used and that it conforms to the specification. So we want to ensure that it is accurate and conformant. B is incorrect. Although a side-effect of well-tested data might be that customers use trip planners, that's not why we test data. As I said, we test data to make sure data is accurate and works. C is incorrect because even schedules that change on the fly must still be tested. And D is incorrect because testing must be done. It does have an effort involved, so we do need to put the effort in, and it has a better net result.

Scott Altman: And now we've come to our fourth and final learning objective, and this is to illustrate how an agency implements GTFS. Our key points are GTFS lifecycle requirements and strategies, procurement language for generating GTFS, making GTFS feed files available for internal and external applications, and describe the use of data by downstream users. We will close with a case study.

Scott Altman: In discussing how an agency implements GTFS, we'll start by discussing the data lifecycle of a GTFS feed. The details of this process depend on the agency, including staff resources, existing software, and the type of operations that the agency has. The lifecycle is an endless process because GTFS must be updated whenever there's a schedule change. The lifecycle process also begins with collecting data. This assures having everything you need to build an accurate GTFS feed. Once we have correct data collected, we can create GTFS. We've already discussed the tools that can be used for this, and we do this at this step. We've highlighted the importance of testing data, so once data is collected and created, we always have to test it to make sure it's conformant and accurate. If there are problems in accuracy or problems in conformance, we need to go back in and recreate GTFS until it is correct. Finally, once GTFS is created, we publish it and make it available to downstream users. We have a final step in this lifecycle-- or a final kind of flow in this lifecycle, and that's the most important one, and that's as schedule updates occur, we need to go back and redo this process over. The GTFS development process is an endless lifecycle because schedules will always change. It may be once a week, it may be once a month, it may be a few times a year, or it may be once every year or every few years, but the cycle does repeat itself from time to time.

Scott Altman: So here we will discuss some lifecycle requirements and some strategy for implementing lifecycle, and the requirements. The first requirement is to update GTFS as frequently as scheduled or updated. This is at least as frequent as the schedules change but there might be other data-- for example, if you have a fare change, you might want to update GTFS feeds. Really, you want to make sure that the information being provided is as fresh as possible. A second lifecycle requirement is to design the implementation lifecycle to work within the agency's processes. So if the agency-- if an agency has fast resources to produce GTFS, we want to leverage that. If they have software, we want to leverage the software. We want to make sure that this is something that is reachable for the agency. It shouldn't be too much of an extra burden to get the job done. The final requirement is don't forget to test, and we've already highlighted why we need to test. Some other strategies is to leverage existing agency systems and processes. If an agency has a scheduling system that can export GTFS, you probably would not want to purchase a whole new system if you already have the system. And if you're already producing some other type of static information feed, you might want to see if you can piggyback a GTFS feed onto that feed. You also want to leave sufficient time to publish schedules and, as I said, it's good to push new schedules out about two weeks at a time. This varies from agency to agency. And you also want to make sure that you have enough time before you publish a schedule to create the data, create the schedule-- collect data, create the schedule, and test it.

Scott Altman: So as I said, when possible we like to leverage existing systems, but this may involve a procurement process. You might need to purchase an add-on for your system; you might need to purchase a whole new tool that can help you generate GTFS; and you might even want to purchase a contractor to do the GTFS production process for you. So in any case, procurement processes often are involved. Fortunately the GTFS specification is fairly consistent so if you tell a contractor to export GTFS or you need to software to export GTFS, that process is generally well understood. High-level requirements might be sufficient for common needs, and if you have a more specific need you could address that in procurement language. It's important and recommended to use a systems engineering process to ensure you are procuring the system that best works for you. You do this by starting with a concept of operations, or a ConOps, and in that you define user needs. For example, we have a sample user need that says, "Agency staff need to be able to export schedule information in a common format." User needs translate into requirements that could be put directly into a procurement document. For example, "The system shall export a GTFS feed which at minimum includes the following files," and it says the files that have to be included right there. If you look in your Student Supplement you'll see some more examples of GTFS procurement language, and it is a fictional example that is there but it should give you an idea of what you might want to say if you need to purchase a system.

Scott Altman: So now we have talked about the lifecycle, how we test and procure systems. Now we'll talk about how we provide GTFS data to the public. Typically GTFS feeds are made available to users of the feed by placing them in a fixed or static location somewhere on the web. Sometimes this might be a private location that internal users use, and that could be different from where the public gets the data, but generally you have one spot where data goes. It is a best practice not to change this location. You want this to generally be the same, and you generally keep the file name of the zipped file with the GTFS the same from feed to feed so that-- a downstream application might set the system to always go to the same location and not change from time to time. So that's why you keep that consistent. Because of this, if you have users who download GTFS feeds and use them for different archiving purposes, it's important to obtain data as frequently as possible and keep it organized on the downstream system, if you need to have one GTFS feed compared to the next GTFS feed. GTFS feeds typically have either restricted or unrestricted access. If access is restricted, users who want to access GTFS data must register and get a credential. They might need to agree to some sort of a license agreement for this purpose. For example, a developer who develops software that uses GTFS might need to update their application within a specified number of days of it being made available. Some agencies don't have restricted access but they may still have a license agreement that it is implied that you're consenting to when you download it. Internal users, as I said, might have a different location than the public, but in any case it is important to notify users when data is available. While some applications that use data might check on a regular basis and be able to fetch this data automatically, others might require some manual process to get data, so you should always tell the users of data when it is available. Agencies also might make GTFS available to a prominent third-party application such as Google Maps, that we've discussed. Such services might require the agency producing the data to agree to a license agreement and various terms and conditions, and they may have to make a commitment to keep the data up to date. Additionally, there are third-party repositories that collect GTFS from a wide variety of agencies to make it available to users in one place.

Scott Altman: Finally, we discuss some applications that are using GTFS, and as we said, understanding users is important and here are some ways that the data is used. The most familiar use of GTFS is customer-facing applications. This includes trip-planning tools, timetable generators, and other tools that the public seeks. Other tools and uses include transportation planning and analysis. GTFS feeds might be archived. Schedule data might be archived and used as an input to planning models as a reference of the transit network. So service planners might have a use for this data. Other applications within a transit agency, such as CAD/AVL systems or fare collection systems-- they might use a GTFS feed as reference to the baseline data that their software uses. And finally, applications that use GTFS-realtime, the client relies on the static GTFS feed as a reference to baseline data. For example, in a GTFS-realtime feed, that references the stop_ID, which comes from GTFS. So we can't have a GTFS-realtime feed without a corresponding GTFS feed, and we will learn more about that in Module 14, part 2. And a final application of GTFS data is known as the National Transit Map. This is a specific application, but it is an effort by the United States Department of Transportation to build a repository of transit service information from as many agencies as possible in the country. In order to accomplish this, the USDOT is asking all agencies to submit GTFS, which will be integrated into this repository. Participation is voluntary but strongly encouraged. It is anticipated that the data repository will primarily be used for research, planning and analytical purposes. This data will better allow transit to be accounted for as part of the nation's transportation network. At the time of this module, this effort is in its early phases.

Scott Altman: We now close with a case study of an agency that is actually implementing and using GTFS.

Scott Altman: This case study is the Westchester County BeeLine System in New York. This is a suburban bus system located in the country immediately north of New York City. They began developing GTFS over six years ago. To facilitate production of GTFS feeds, they purchased an add-on to their Trapeze FX system, which is their scheduling system. Before this, they had a system for disseminating data but it was a proprietary ASCII CSV format. ASCII is a difficult-to-read, difficult-to-process format that represents letters and characters as a series of numbers. It wasn't a widely accepted way of providing schedule data, so they decided to purchase a way to create GTFS. So the GTFS process is coordinated by a member of their information technology staff at the county's Department of Transportation. So once the schedule is entered into the system, that process takes about two weeks, the IT staff member verifies the accuracy of the data. Once that's done, the exporting process really only takes 15 to 20 minutes. Because their scheduling system only updates the feed for the future service-- so if they're creating a new service that's starting in a few weeks, their system only exports that. But as we said before, downstream applications often require current service to be matched with future service, so they run the data through the Google Merge tool that we discussed. This makes it able to be immediately located into most transit scheduling systems. Additionally, exceptions for holidays have to be manually added to the feed, and that's a relatively straightforward process that takes a matter of minutes. Once GTFS feed is complete and tested, it is made available to downstream users. A key user is the New York State Department of Transportation's 511NY Transit Trip Planner. That New York State Department of Transportation, or NYSDOT, has a consultant that performs this testing to ensure conformance to the specification and provides support to the agency to test the data and make sure it's accurate, as we discussed the importance of testing. At this point, data is loaded to the trip planner and it is also made available on the 511NY developer page to third-party users. Currently BeeLine does not provide the data to Google Maps, which although is the most popular trip planner, they haven't yet done that. They are considering that. But they do provide data to only third-party sources through this third-party 511NY developer page. Over the years, they've encountered some primary issues. One was related to stop times occurring with an illogical sequence. In most cases there was data issues that were found in the scheduling system that had to be resolved, and once resolved, then they could fix the problem and have accurate data. So this case study shows that it is easy for an agency to integrate GTFS production into their workflow, and two, that quality control is critical to catching problems early and ensuring the data is correct.

Scott Altman: We have our final activity.

Scott Altman: And the question is: How should GTFS feeds by made available to downstream users? Our choices are, A, GTFS should not be made public; B, they should be written on a CD and mailed; C, they should be printed on paper; or D, placed in a fixed location on the web.

Scott Altman: Let's look at our answers, and the correct answer is D. GTFS should be placed in a fixed location on the web, and that's how it should be made available to downstream users, and the location should be fixed and accessible. The other choices are incorrect. A is incorrect because GTFS is intended to be made available to downstream users. So it should be made available to the public. B is incorrect. We would not write GTFS on a CD and mail it. That's an inefficient process in this modern technological era. And C-- we can't print GTFS on paper because it wouldn't easily be incorporated to downstream systems, and that is a non-efficient method.

Scott Altman: So here's a summary of what we have learned about the GTFS specification. One, GTFS is used for disseminating static transit schedule data. Two, the GTFS feed consists of a series of zipped text files that define aspects of fixed transit schedule data. Three, GTFS feed must conform to the GTFS specification and the feed must contain accurate data. And four, the GTFS feed should be integrated into a transit agency's existing processes. So overall, this module taught us about how GTFS feeds are created and used.

Scott Altman: So at this point we have completed the module. We thank you for completing this module and learning about GTFS. We ask that you please use the feedback link below and provide us with your thoughts and comments about the value of the training and give us your feedback about if it was or was not useful to you. So we thank you very much for your participation today and learning with us, and have a great day.

End of 2016_08_18_14.34_Mod_14_P1_Final_Record.mp4