# Inpatient Quality Reporting Program

## Support Contractor

**CMS QRDA Submission Errors Eligible Hospitals/Critical Access Hospitals Presentation Transcript**

**Moderator:**

**Debra Price, PhD, MSPH, MEd**
Education Coordinator, FMQAI/HSAG

**Speakers:**

**Rick Geimer**
Lantana

**November 5, 2014**
**3 p.m. ET**

**Debra Price:** Hello, and welcome to the webinar. Thank you for joining us today. My name is Debra Price, and I am the host for today's event. This slide shows you how to use our Question & Answer (Q&A) feature for the event. Follow the directions on the slide in front of you. Move your mouse over the WebEx navigation panel at the top of your screen. It's a little green navigation panel, and a menu will drop down. The menu looks like the menu on the right-hand side.

Click on the Q&A icon. The Q&A panel will display on your screen. And then, in that panel, click the dropdown arrow next to "Ask," and select "All Panelists" if you want all of our subject matter experts to answer your question. Type in the question you want in the space that says, "Type questions here." A subject matter expert will view that question and will answer it. Click "Send" to send that question.

And now, I'd like to introduce our guest speaker for today's event, Rick Geimer. Rick is the Chief Technology Officer at Lantana Consulting Group. He's the co-chair of the Health Level Seven

(HL7) Structured Document Working Group. He is an HL7 Clinical Document Architecture (CDA) Release 2 (R2) Certified Specialist, and he's the co-editor of Consolidated CDA and other specifications.

Rick, take it away.

**Rick Geimer:** Great, thank you, Debra. Can you hear me okay?

**Debra Price:** Yes, we hear you fine.

**Rick Geimer:** Okay, great. Well, thanks all for joining. As the title says here, this is going to discuss some common Quality Reporting Document Architecture (QRDA) Submission Errors, in fact, the top ones that are commonly received, and walk through the issues themselves and potential solutions for them. And then, the last piece of this presentation, I want to talk about some common debugging approaches that you can take before submitting files to basically avoid a lot of these errors before they even get to the Centers for Medicare & Medicaid Services (CMS).

Next slide. Alright, so we'll go through some of the common errors first. I meant to have another slide here listing them, but I'll just kind of speak to them. The errors basically will be ones regarding document format whether it's actually a valid QRDA that's being submitted, issues with the CMS certification number or CCN, issues with encounter, admission, and discharge times, some patient metadata problems around ethnicity, codes and patient IDs and such. And then some, I'll say peculiarities, about QRDA or its parent CDA which is needing a legal authenticator and text for sessions, human readable narrative text. With that I think we can start digging into the first few here.

So, next slide please. Okay, the first one and probably the most common one is the QRDA document format error. So the error message that's on the screen is what you'd get back when you fail to submit a valid QRDA document. And what this basically means is that a lot of people are submitting things that are not QRDA files. Perhaps they're submitting Microsoft Word documents or other things of that nature. So it is very important that you actually submit a valid QRDA file.

So, next slide. Okay, so basically what I just said there: any documents that are submitted to the QRDA system need to be valid QRDA. So if you submit an MS Word, an HTML file, a PDF, and so forth, then it will be rejected. That would be the most common error: people just not actually sending a QRDA. In order for it to be a QRDA document, it needs to be compiled with the Schema that's provided by the HL7 Quality Reporting document architecture specification. And I've got some slides later on that [and I'll] talk about it.

So, next slide here, yes. So the correction for this one is just to submit a valid syntactically correct QRDA file. And when I go over the common debugging approaches at the end, I'll actually go through step by step on how to download the QRDA Schemas, associate them with the QRDA document, and actually validate that you've got all the syntax of QRDA expressed correctly. So we'll get to that solution [in] a little bit because it's a little bit more involved than the other ones.

Next slide. Okay, another one that comes up quite typically are validation errors with the CMS certification number or CCN. Some of the common messages that are reported are above: CCN cannot

be null; the represented custodian organization shall contain exactly one ID such that it contains basically a CMS certification number; the extension shall […], and you know things of this nature pop up. In other words, there's more than one error message that can occur depending on exactly how the CCN is expressed and where it's placed.

So the basic meaning of all this is that you need to submit a valid CCN. It needs to be in the right location, and it can't be a demonstration or a testing CCN. It has to be one that's live for your organization.

So, next slide. So here's some examples of some mark ups, some QRDA XML that will actually invoke these errors. The first one, the CCN is missing entirely. So we've got a custodian element with an assigned custodian and a represented custodian organization. This is sort of the nesting that occurs in QRDA documents for where you would know to be the organization that is creating and maintaining this quality reporting document architecture file. And, there's a requirement that you actually pass in the CCN right where those three red dots are. So in this case, this sample is submitted entirely. So therefore, it's going to throw an error. The second example has what's called a Root Object Identifier (OID), or this is an OID. Just so folks know, it's a string of numbers and dots that basically identifies that this is a CCN.

However, it is missing the actual CCN that wants to be reported. In other words you put it on an ID and you said this is a CCN, but then the actual CCN hasn't been provided. It's like having a field that says, "Please fill in your driver's license number" and leaving it

blank. So yes, you know that a driver's license number should be here, but you don't have one.

The last example is kind of a tricky one. This is one that you see when lazy developers get involved. They copy the sample files and then paste them, literally, in their programs, and output them exactly as they were in the samples. And an example QRDA would have a CCN of 800890. But that's not actually your organization's CCN. So this is an example where you'd be rejected for actually sending a demonstration CCN instead of an actual one for your organization.

The next slide. Okay, so the way to solve this is to make sure that inside the represented custodian organization in your XML file you have an ID element present with that root attribute ending in that OID, or that string of numbers and dots, that ends in 336. And then you have an extension attribute in there as well, that contains your CCN. So I put that in green and italic to make sure that you actually need to register with CMS, get a CCN of your own, and put that in there. So every organization should have their own CCN. You cannot use the one that's provided in the example files. You actually need to populate this with your own content.

Okay, so let's go ahead and get to the next slide. All right, so moving past the CCN errors, we're going to talk about another common law which is admission and discharge time errors. So these are errors where you might get something that says the patient shall contain a discharge time report on the encounter that is precise to the seconds either for admission or discharge. This is a very common error that comes up. What this means is that admission and discharge times must be present and must be in the

form both listed below, which is basically a formula which means the four-digit year, a two-digit month, two-digit day, hours, minutes, and seconds. And then, plus or minus, that should actually read a UTC offset, or basically offset from Greenwich Mean Time. So if that format is not present, you'll get an error. If you haven't provided enough digits in your times to actually be accurate to the second with a time zone offset, you can get the error, as well. One other error that you might get is that if the admission time is later than the discharge time. In other words, you're saying they showed up after they left. That's also another big "no-no."

So let's go ahead and go onto the next slide, and we'll talk through a few of these specific errors and solutions. Okay, so here's an example of admission, basically where both the admission and discharge time are invalid. In this case, someone output a low element which is where you output the admission time. And that's basically … Just to explain a little bit, an encounter has an effective time that can have a low and high value. The low value is when the encounter started, and the high value is when the encounter ended, and those equate, basically, to admission and discharge time. So in this case the low element is present, but there's no value on it. In other words, you're not actually stating what the low time is. So this will come out as not having sufficient precision basically because there's nothing there at all. The second one, we've actually populated a value, but it's only precise to the day. So this one is basically saying on February 27th of 2014, this encounter ended. But you haven't given a time for it, so no one knows whether this was a one-hour encounter, was this an all-day encounter, or what? So, that's why it's important to include the precision that this requires so that we can actually calculate properly when this encounter started and ended in the length of time.

So, next slide. Okay, so this one has a partial correction. I note that both these times are now. They meet the formatting requirements, okay? So they're precise to the second. They actually have the UTC offsets so you can tell what time zone this occurred in. However, it's a little bit tricky to note this on just glancing at it, but the admission time is on February 28th, and the discharge time is on February 27th. So the discharge actually occurred a day before the admission which is another "no-no." So, this one will still fail. Even though you've got the syntax right, it's semantically wrong. You can't arrive after you've already left.

The next slide. Okay, so this is one where now everything is finally correct, and this should be accepted where we've got an admission time that was, in this case, on February 25th, so two days before the discharge time. And, give or take a few hours there because the hours are different, the format is precise enough. It's precise to the second, or at least padded out to meet the format there, and it contains the UTC offset, so you know when this occurred and what time zone it occurred in. And everything should be processed correctly at this point.

So let's go ahead and move onto the next slide and the next topic which is Patient-Related Issues. Okay, one of the common ones that shows up are errors with the patient ethnic group code. This is a requirement in this country, to actually pass in information about race, ethnicity, and such. And so, it's expected that there be a code such as these present in your QRDA files. It is a little bit odd that the ethnic group code is a very small value set that really just contains two codes, one for Hispanic or Latino, or one for non-Hispanic or Latino, but these are required to be present in these files. And, there's a link in these slides that actually goes to the full

definition device with a little bit more explanation about it and the codes in it. But the main reason why you get this error is if you've omitted these codes or perhaps you've used incorrect codes. There are other value sets for calculating these, but only one of them is actually correct for QRDA files for being submitted to CMS.

So, it is also possible to note that there are cases where a patient will refuse to state [their ethnicity]. It is possible to express that using what's called a nullFlavor. So, there's a syntax for doing it, but there's also several wrong ways to do it. So the next slide we'll go into some of the ways it will show errors, and then I'll follow up it up with another slide that shows how to correct this.

So next slide please. Okay, so in this case we've got a couple of pieces of data on the patient; one is the race code, which is present, and then it's followed by a guardian. And, if you notice there's no ethnic group code in there at all. So this is a case where that's actually missing, and that'll just throw an error because you haven't stated the ethnic group at all in this case. The second example below has an ethnic group code present, but there's no data in it. So this is one way that folks may think is an easy way to say [they] can't state this or don't have the information, but it's actually not legal syntax in a QRDA document.

So let's go ahead and go to the next slide. Okay, so this first example here shows how you'd actually expect to pass in data for an ethnic group code. So, it has the actual code of 2186-5, which is non-Hispanic or Latino. There's a code system match to be present which is an OID, as I mentioned before, a string of numbers and dots. In this case, that code system represents the CDC race and

ethnicity code system. So this is all valid, legal data that you can then send in a CMS code process.

But what if you actually wanted to do what was in the second example on the previous slide and not pass any information? Perhaps it's unknown. Perhaps the patient declined to state, etc. In this case you would use, I like to think of it as an escape hatch in the QRDA standard, which is using a nullFlavor. And I'll use an example here on ethnic group code, but it could apply equally throughout many parts of a CDA document. If you don't have a particular time of admission or something that's unknown, you could put in nullFlavors there, as well. So what this does in this particular one, "NI" just stands for "no information." It's the most generic nullFlavor that HL7 allows you to use, and basically just states there is no information, or it's just a default null. There are other ones that are more specific for "Unknown" or 'Declined to State" or "Masked for Reasons of Privacy," and such. These are all available in the QRDA specification itself, along with guidance on how to use them all. I won't go through all of them there or on this call here, but I do want to alert people to nullFlavor; let them know that this is the proper way to identify when information is unknown.

And, if you have any question about which nullFlavor to use, the most generic one of "NI" is almost always appropriate. It's the parent of all the other nulls. And so, you could sort of default to that if you don't have something more specific to say.

Next slide. Okay, one other common one that shows up on patients is missing IDs. An example error message on this one would say the patient role shall contain one ID and possibly some other things, and it should be a Medicare HIC number indicator. So, what

this means is that the patient ID was either not specified at all, or possibly that it was specified but in the incorrect location. Some other options are it was specified without a proper OID to scope it and basically identify that it's a Medicare HIC number.

So let's go ahead and go to the next slide. All right, so this is the arrow up at the top, the example at the top there shows a patient role. And it immediately starts with an address, but there's no ID present there. Now the way the QRDA Schema works is order is very important. So an ID for a patient, or the patient role, I should say, actually needs to appear before an address occurs. So in this case that's not being seen. That's not present. So this will actually fail validation against the QRDA Schema. If you didn't have it in there and it's required, or rather if you had one, and maybe you put it after the address. In this case, we're just assuming that someone's omitted it entirely. However, the way that this is structured, there are actually two legal places in QRDA where you can put an ID. The first one is under "Patient Role" and the second one is under "Patient" itself.

And I'm going to pause for a second and describe a little bit of information about HL7 QRDA and its parent CDA work. And actually, the HL7 Reference Information Model (RIM), in general, is that basically everything you see follows this paradigm of participation, role, and an entity.

So the element at the top that's called "Record Target" is basically the participation. You're saying that this, I'm going to identify the person that's participating as the target of this medical record. Next you would state their role. In this case, their role is the role of

patient, and you would have information here that is applicable to that role such as a medical record number or something like that.

That kind of ID is specific to their role of patient. Then lastly, underneath there, you actually have a “Patient Entity,” which is where you would put information that is specific to the patient themselves. It doesn’t matter about the role. So, things like their driver’s license number, their name, those are properties of the person itself, things that they carry with them when they leave the hospital.

So that’s why we have this separation here, and why when you’re trying to transmit the Medicare HIC number it actually needs to be under the “Patient Role,” in this case. So that’s the way the QRDA specification has said that this [is] something applicable to someone in the role of patient, and that it should be moved up to there.

So the next slide? So, this shows the solution of this particular error, which is you would just move that ID up into the “Patient Role.” And again, it needs to occur before the address element. There are other elements that are legal under Patient Role in the Schema, and, I would say, refer to the Schema to see everything that’s possible. But generally, this would be the solution here, just to put the ID right underneath the Patient Role, and for most cases that’ll work just fine.

Okay, next slide. Okay, now this one throws people a little bit. One thing about QRDA that it’s important to understand is that they are documents, and they are clinical documents. They are patient-

specific, and as such, they are similar to just about anything else that you could find in a patient chart. It has patient identifiable information. It has information about their health and such. And like other clinical documents, they need to be signed, just like if you were to go into a hospital and have an encounter and a clinician filled out some information that was going to go in your chart. They typically would sign that at the end. So QRDAs are supposed to be authenticated. They're supposed to have a person who is attesting to their content.

So, every time that I mention this to developers they say, "Well, I'm creating these automatically from a system. What do I put in here?" And the answer to that is, you take the same approach that you would for any other legal document that is created by computer, which is you typically would put in someone in a position of authority, like an officer of a corporation or such, in there, the same way that if a corporation with 10,000 employees prints out paychecks every two weeks. You'll notice that somebody's signature is on all those paychecks, because checks need a signature. It's not that that person has actually gone in there and physically looked at and signed every single check, but they are taking responsibility, in their role as an officer of the corporation, for the content of that check. So, it is basically a way of stating that someone is responsible, and it is important that clinical documents have some traceability of responsibility. So that's why these are required to be in here.

So let's go ahead and go to the next slide and show some common examples that would generate the errors that people see around "Legal Authenticator." The first one is when the Legal Authenticator is just missing entirely. This is the tricky one because the QRDA

Schema itself, or the CDA Schema that is distributed with QRDA, won't enforce this because there are some use cases where clinical documents need to be released for patient care before someone can go back and sign them later that evening. So there's the requirement that there are some cases where unauthenticated documents are created temporarily. But in general, when they're done they're done, and they should be signed. QRDA basically requires the signature at the point where you submit it to CMS. So, in this case, the legal authenticator is missing. It should be appearing right after "Custodian" in this case, and it's not.

The next slide, Okay, so here's a case where someone just decided to try to … Maybe they got that error that said "Legal Authenticator is Missing," and they tried to put in a legal authenticator element but didn't populate it with any data. This one will fail typically because a "Legal Authenticator" has some required elements underneath. Typical things are: the time that someone authenticated this document; the name of that person; the organization that they come from; things like that need to be present.

Next slide, okay so here is a big complete "Legal Authenticator." As you can see, there's a lot of information in here. I'll go over a few of them. The first of them is the time, in this case. The time value here is the time that someone actually signed this document. It has a signature code of S. This is just sort of boiler plate in QRDA. It basically says that this is a signed document. Then we have "Associated Entity," which has that person's information, such as their ID and their address, telecommunication information, and then, the person themselves which has their name in this case, Henry Seven. So this is basically the information that you need to pass as the "Legal Authenticator." Again, this could potentially be a

person in a role at an organization versus having someone review every QRDA and signing them individually. One other important thing to note, this is not a digital signature. Rather, this is the contact information for the person who is taking responsibility. So if developers are on the line, they're asking is this a digital signature? No. If you want to digitally sign QRDA documents, that's a different step, but it doesn't remove this requirement to actually include the name and contact information of the person. So basically this is so that anyone receiving this would know who to go to if they had questions. Okay? That person may delegate and say, "Well, yes, I'm an officer and I don't know specifically about this QRDA, but here's the person that does, and here's their phone number." So, it is the go-to person for any questions and issues about this document.

Okay, let's go onto the next slide. Okay, the last one that's again something that confuses people about QRDA documents is that in addition to being documents that have computable data that is sent from machine to machine, they are actually meant to be readable by humans. So every QRDA document has to have what's called a "narrative block," okay, which means that every section in there needs a narrative rendering of the content in there. This is a failsafe basically that if machines are down, systems are off, someone can still take this document, bring it up in a web browser, and read it and not have to understand what code 3768953 means. And I just made up that number, so don't look it up. But they could actually read what content is being reported in a human, readable fashion.

So every QRDA document must have a "narrative block" in each section that conforms to the rules of the QRDA specification, and

there are a few rules that throw, particularly software developers, for a loop. So I'll go over some of them in the next slide.

Let's go ahead and advance the presentation. Okay, so here are three common examples that you'll see with section text. The first one at the top just has the text missing entirely. Now this is another one that the clinical document architecture or CDA Schema will not check for you because there is a use case for some clinical documents where you have a section that contains no content itself. Rather, it is just a parent of other nested sections. In other words, [it is] a section with subsections, but no text of its own. So for that use case, you'll actually see text as optional in the Schema, and a lot of implementers who only look at the Schemas and don't read the rules might look at that and then say, "Well, that means I don't need it." Well, the case is, for QRDA it is required, and we actually don't promote the use of subsections, and require every section to have text present. So, you'll get an error if you try to submit a document like the one at the top.

The second one below is another one where people say, "Okay, well I'm just going to put in a text, but I'm not going to put any content in here." Again this might fake out some validation systems, but it actually still breaks the rules of the specification that says, "If you're going to have any content in here, any computable content, there needs to be all the clinical relevant stuff present in the text for human readability as well." So, that needs to be present. You can't just pass on an empty text; you actually need to put the text in there.

So the last one, at a first glance, [appears to] look great. You look at it [and] you say, "Great, they've got some patient data in there.

They've got some mark up. This all looks good." And what you'll notice, though, is that in this case, they've used HyperText Markup Language (HTML) HTML markup or Extensible HyperText Markup Language (XHTML) markup. And it's unfortunate, but the CDA specification does not allow raw HTML. It does have its own markup that is present in text. It's very similar to XHTML, but it is not, with the exception of the tabled model, equivalent.

So let's advance to the next slide and we'll talk about the solution here. Okay, so in this case, it looks very similar to that last option. The only issue is that if you were to bring up the Schema that's distributed with QRDA, you would see that inside text you don't have a "P" element that's allowed, but rather one that's called "Paragraph" So, some of the tag names, again, are different there.

The content they allow is a bit more restricted, and this is to make sure that your clinical documents are exchanged, not with a thousand website bells and whistles and possibly embedded flash movies and things like that, but rather, a very straight, bland, clinical content that you would expect to see as if you were picking up a document in someone's patient chart. So it's just the data that you need to communicate for human readability. So that's what you'll see present in here. In this case, we've got a paragraph tag, and that would pass the validation and allow you to get on with your submissions.

Okay, so let's go ahead and move forward now to the second part of this presentation, which has how do you avoid these errors, or at least a lot of these errors, in advance of submission. So I'll talk about some approaches that let you catch these errors yourself. It won't catch all of them, specifically things that are CMS-specific,

like formatting of CCNs and look-ups to see that you're using a CCN that's been registered, but they'll let you get past what is probably the vast majority of errors, which is document format, and counter times that are improperly formatted, missing codes, things like that.

So next slide. Alright, so the best way, again, to prevent submission errors, is to validate your documents before you send them. And there are two primary steps that all implementers should perform. The first one is to validate against the QRDA XML Schema that is provided from HL7. There are tons of XML tools out there in the world that validate XML Schema. It's nearly universally supported by all XML tooling. So there's really no excuse for not doing that piece before. And that actually, if everyone would just validate against the XML Schema, that would eliminate the top error, which is the incorrect document format error. You would find that out right away, that you're not sending a valid QRDA file.

The second one, which is a little bit more complicated, is validating against the QRDA Schematron schema. So Schematron I'll describe very quickly. It's another schema language for validating XML files. It is an International Standard for Organization (ISO) standard, and even the creator of Schematron kind of describes it as a tool for cleaning up the dusty little corners that XML Schema itself can't reach. So XML schema is a broad broom, so to speak, for validating and cleaning up files, but it can't get all the little corners. So, Schematron is really the Dust Buster of the validation world. It gets to all those little places and helps you clean the data even more so than it would be normally.

So next slide. Okay, so the first question is, “Well, where do I get these?” The easiest place is to go to the HL7 website, and that stands for Health Level 7.org. HL7.org is the site, and there’s a page there called “DSTU Comments.” Now DSTU stands for Draft Standard for Trial Use, and that’s where you’ll actually find the QRDA specification. So if you go to that page and search for QRDA, you’ll notice that there’s a little search box in the center of the screen there. It’ll filter it out and show the various QRDA implementation guides that are available. The one you want is the second one that says *Implementation Guide for CDA*. [It] relates to quality reporting document architecture Category One, DSTU Release 2, a very complicated name, basically that means QRDA Category 1, the second release of it. You would then click on the download link and that would give you a ZIP file that would contain the QRDA specification, the QRDA standard, and all the supporting files that are provided with the standard itself.

So next slide, so once you get that ZIP file and extract it you’ll end up with a folder that looks like the picture on the right. So this folder contains a bunch of information. The key ones here are the specification itself so you can actually read the rules of the standard. There are rules in here that are probably not expressed fully in the CMS guides. I know there’s been an attempt to get a one-stop shop, but it’s good to have this handy so you can actually see what the standard says in case there are any questions there.

Some other key things about this folder once it’s extracted, or the ZIP file once it’s extracted, is there are three subfolders in there: there’s one that says Schema, which is where the QRDA XML Schema that I’ve been touting is present; there’s one that [says] Schematron, which is where the Schematron schema lies; and then

there's an XML folder, and that is where you'll see sample files and such.

Okay, the next slide. Okay, so if you want to use the QRDA XML Schema you would actually drill down into that Schema folder and you notice that it is several levels deep. You would actually open up Schema and then CDA and then Infrastructure, and then another folder called CDA underneath. I know that sounds kind of strange, but that's related to the fact that these Schemas are actually generated by HL7 tooling from models. So these are basically modeled first, and then Schemas are generated from those models. And some of the structure is related to the fact that this is sort of tying back to a model called the "Reference Information Model."

I won't go into details about it, but bear that in mind when you think about, "Why is this Schema structure a little bit strange?" It's because there's a lot more stuff going on behind the scenes. But, the important thing to note here is, that if you do move this Schema around, it's not sufficient just to move the main driver file, which is CDA_SDTC.xsd. Okay? If you were to actually open up that file in a text editor, you'd find it's about five lines long, and all it does is it references several other files like that POCD_MT blah, blah, blah 040 file, and some other files in other directors. So, be aware that it's not that file that is the Schema, but rather that whole Schema folder and all the nested and cross-referenced directories within it that needs to be moved around. Your files will point to this main driver file which I'll show on the next slide. But be aware that that whole Schema package needs to be moved as a unit if you move it around.

Okay, next slide. Okay, if you want to associate a QRDA file with the Schema, you do this using the rules of XML and the XML Schema specification. This is not an HL7 spec or a CMS spec, this is a Worldwide Web Consortium, or W3C specification. What you would do here is, you would create an XSI:schemalocation attribute on clinical document, which is the root of the QRDA. And then you would associate. There are two parts to the content there. The first part is urn:hl7-org:v3. That's the name space of the Schema, and then there's a space character, and then the actual path to the Schema itself, okay? This is meant to be a relative path from wherever your QRDA file is stored. So what I have up here is just an example. If you move all your Schemas into C:QRDAstuff/Schema, then that's what would need to be present if you had your QRDA files somewhere else. So, just be aware, that needs to be a relative URL from your QRDA document.

Okay, next slide. Okay, once you've got that link, you can open up the file in any number of XML editors that are out there and validate it. I won't go into specifics because every tool has their own way of validating XML files. You can validate them from a command line. There are online validators out there for CDA and QRDA files, as well. But this example here, that I'm showing, is just in a desktop tool called Oxygen XML. Again [I'm] not recommending this tool. It's just one of many, but I needed something for an example. So what you could do then is, after you bring it up in here, you could click a button to validate this, and then you would get an error report generated with any conformance issues with this particular XML file. This one in particular had an error with the language code where it's in the wrong place. It actually should have been above the effective time instead of below. I know that's a little bit hard to read on the screen here, but these kinds of things are important

and will actually cause you to fail with that first most popular error, which is the Document Format Error, if all these are not corrected. It takes a little bit more time to walk through and correct all these errors, but it is time well spent because your files will be rejected outright if you don't do this.

So this is the biggest thing that folks can do to cut down on the QRDA submission errors, is just validate against the QRDA XML Schema that is provided by the standard from HL7 before you submit any documents. And, in fact, I would even recommend that you not try to submit a single one until you've validated at least a dozen of your own files that are generated from your system. People who are really aggressive with this and want to take, probably what I consider to be the best approach, would actually implement a validation system on their sending side, which would not send any files that don't validate against this schema, and rather put them into queue for human review for developers or other folks to take a look at it and say, "Hey, there's something wrong with this file," because even if you send it anyway, it would just be rejected.

Okay, let's go ahead and go to the next slide. Okay, so let's say that you've got past that first part. You've associated your Schema's. You've validated a bunch of files and they're coming out clean without any errors. You think great I'm done. I'm going to go ahead and send this off. Well, the second issue that you'll find is all those other errors that are not Document Format Errors, many of them would have been caught if you ran the Schematron Schema.

Now, a Schematron is a little bit more difficult. Not every XML tool out there supports it, so there are a more limited set of tools for

validating a Schematron. And frankly, a lot of developers are scared of it because they maybe haven't heard of it before. But really you shouldn't be [scared]. It's fairly friendly, and I didn't put it on this slide ... I probably should have, but note that Schematron Schemas basically compile to Extensible Stylesheet Language Transformations (XSLT), which is a very widely supported XML transformation language. And when you run that XSLT against a QRDA file, you actually get an XML report that says all the errors. That's actually what nearly all the online validators will do for you, is they'll just go ahead and run a report, generate an XML file, and then show it in a browser with the style sheets so it looks like HTML. But Schematron is actually not difficult at all to figure out and to learn to use, so don't be scared of the word if it's unfamiliar. Just go to Schematron, I think it's Schematron.com, in this case, or Google for Schematron, and you'll find that page. But again, it is an ISO standard. It is fairly widely used, again, just not as widely implemented in tooling as W3C's XML Schema.

So, a long winded description there, but basically to use the Schematron Schema that's provided with the QRDA specification, you would open up the Schematron folder and you'll see two files in there. One says QRDA Category 1 Release 2.sch. SCH is the extension for Schematron that we're using here. And there's another one that says voc.xml. It is important to note that these two files need to be moved as a unit. There is no big nested directory structure, but if you just move the QRDA Schematron somewhere and try to run it, you'll get errors because that file does reference the voc.xml file which is a vocabulary file. It has all the terminologies and value sets in there that are needed when checking a QRDA file. So an example would be in here you would expect to see the two legal codes for Ethnic Group Code, and that's

how the Schema could check that you're actually using proper codes for that and such.

And so let's go ahead and go to the next slide and talk about how to associate your documents with it. So, the W3C even though it's an ISO standard, the W3C wrote the suggestion for how to associate Schemas like Schematron with XML files. And so, HL7 and CMS, in this case, have just used their recommendations, which is using an XML model, what's called a processing instruction in XML speak. And you would then note the path to the Schema, the type is an XML file. Schematrons are written in XML. And then you'd actually have in a name space that declares the type, in this case this is a URL for an older version of Schematron. In this case there is an ISO version, but the ones that are currently being used, I believe, have this older Schematron 1.6 URL here.

Next slide. Again, you can bring this up in a tool. The only reason why I showed Oxygen in this case is because it is one of the tools that supports both XML Schema and Schematron, so it allowed me to use the same tool for both examples. Again, not a recommendation of the tool itself, but I needed to show something.

So this one [in which] we've taken a QRDA file. You can actually open it up in an editor or run it through any command line validator or send it to a website that does Schematron validation for QRDAs. And when you run a Schematron Schema, or if you develop your own system for processing them, you need to know that the Schemas provided by HL7 have what are called *several phases* built into it; one is for "Errors" and one is for "Warnings." Typically what you want to do is choose the "Errors" phase. The "Warnings" basically give you guidance for clinical best practice, but I don't

want to go too much into theirs because fixing every warning that's possible sometimes takes more time than you have. What we'd like to focus on is just reducing number of actual errors, in this case. So, don't worry about getting it warning-free, but do get it error-free before you submit, and that would cut down a lot of processing.

So the Schematron Schema, when you run it, it'll actually go and list a lot of errors if they are present in your file. And what's interesting and again, hard to read in here, and apologies for that, is that the errors that come back will have conformance numbers, at least the ones that are in the Schemas provided by HL7, and these are actual references into the QRDA specification itself. So, you can search on those conformance numbers and find out what rules you're failing and what was expected instead. So, a lot of times with a lot of validation systems, you just get back these meaningless error messages and you have no clue what they stand for. This one actually gives you it pointed right back to the spec and you can go back and see, "Wow! Yes, this is how I should have done it." So very helpful, and I encourage folks to get familiar with Schematron and again don't be scared of it. It's a good thing.

Okay, the next slide. I think we're getting pretty close to the end here. Yes, so I think I'll just hand this back to Debra at this point, and thank you all for listening.

**END**

This material was prepared by the Hospital Inpatient Value Incentives, and Quality Reporting (VIQR) Outreach and Education Support Contractor, under contract with the Centers for Medicare & Medicaid Services (CMS), an agency of the U.S. Department of Health and Human Services. HHSM-500-2013-13007I, FL-IQR-CH8-11072014-12