Standard is a conceptual model if you haven't caught what's going on an OCC.  
  
Recently, we found that the proliferation of different recruiting formats make it difficult to create a single standard for pure Y So what we've done in many cases, we've created a conceptual model, which is a platform independent standard, which is implemented through a set of implementation specifications.

This is a conceptual model of the platform independent standard, which will then be later specialized for different technologies, by an implement implementation of Specification.

You see the conceptual model is captured using UML.  
  
So there is an underlying UML model to the standard.  
  
And what you have in the standard document, is the some image captures out of the UML model, as well as a data dictionary that ultimately will be generated directly from the ML model right.  
  
Now it is still a handcrafted product.  
  
Our goal is to have your ML model to generate the content of the standard front of you a mental model, and that effectively makes a model normative.  
  
And the standard itself is a representation of that normative model.  
  
Some fundamental principles going into this is that appeal is not a dataset, is a feature type.  
  
PoI has to integrate with an existing dataset, which means it has to be able to adapt to any teacher model that you're working with.  
  
So therefore, POI conceptual model has been designed to be very adaptable.

Second of all, don't re-invent.

If there's already an existing class, it's a concept is already defined somewhere, re-use it, don't create a new one, which is almost, but not quite the same, and once again, keep it flexible.

One thing we can do if we do this right is we basically, you can use the UML model and the Standard, to define a logical structure of a pure why.  
  
But the actual, meaning the semantics, can be defined externally.  
  
And we use code list to do that.  
  
There are other ways for doing it, too.  
But the nice thing about this is that I don't have 12 different copies are the same data structure.  
  
Each one means something different, because we gave it a different name.  
  
I can define a data structure once I can encode the processing for a data structure, once, but I pull the meaning of that data structure by a, some sort of tag or identifier,

The relevant standards we're building on, 1 on 103 is the underlying data types that are used across all of the ....  
  
Well, supposed to be used across all of the standards  
  
I'm not going to argue that it necessarily, as ISO one day, and Windows seven is the ISO Geometry model are.  
  
So when I went to an Irish future model, if you've been looking into OTC at all, you know that our fundamental concept is the feature, feature is a real world object.  
  
Features have geometry, so these three standards, 103, when I struggle more than I, pretty much lay out the whole foundations for OTC data, show the RTC data architecture.  
  
We brought an ISO 1 1.5 because scientists metadata that there's a rich collection of data types in there which are very suitable to be re-used.  
  
We're also pull some stuff out of City GML three, which has some extensions on the future model, which are quite applicable to what we're doing here.  
  
So when you look at the POI model, you'll find very little in the ...  
  
Share which is actually new. Most of it is re-used.  
  
Here we have a brief overview of the ISO general future Model Y R, so when our 109, once again, a PWA feature, as such, it inherits all the properties of a future, including the ability of a PWA to be a collection of POI.  
  
It also means that we're re-using the whole concept of attributes, and the properties of features are actually re-used in POS.

Here's Yasser geometry model, this is the geometry model we're using for ....  
  
It's taken straight out of ...  
  
107.

What we do have done is we've restricted the ISO geometry model to only include points, lines, strings, and polygons, And that is something that we need to validate that restricts.  
  
There's that restriction sufficient, or do we need to break the geometry a little bit more complex.  
  
The problem is the more complex the geometry, the harder it is to implement appear.

The less likely it is that the POI will actually serve the purpose of providing a simplified representation of a complex feature.  
  
This is our starting point.  
  
We think this is sufficient.  
  
We like broader feedback and justice word.  
  
So this is your overall POI model.  
  
The \*\*\*\*\* in green are what's actually in the Model.  
  
The rest of it is what we are.  
  
Some of the stuff carried over from the ISO standards are we are only adding 1, 2, 2, 5 new classes.  
  
Abstract feature is pulled out of fog, city GML.  
  
Basically, it's a special, does a better job of articulating what your patient future future model is.  
  
Abstract future of lifespan, Yes.  
  
Or cardinality?  
  
Cardinality gifts from temporal properties to the abstract feature when it was created Windsor valid, things like that.  
  
Abstract pure Y which are our abstract definition of a pure why this is basically this is a P Y.

This clashes of POI MTOR.  
  
Properties is how we map properties from the future data dataset just being described into POI itself.  
  
So what this means is that I haven't actually we haven't actually define fixed static model for ....  
  
We have defined it core model for pure wires and a mechanism to bring additional properties into the POI as needed by your use case and by the data that sense it is supported.  
  
And finally, an important link is feature of interest.  
  
This was borrowed from number of different standards, we see this a lot, and see?  
  
What this is, is this is a URL to the feature in the dataset that this POI is describing.  
  
So this is how we associate a with a feature in the dataset.  
  
This is very important, because as you use a ..., you probably will want to drill down from a pure Y, which is a simplified representation to the actual data underlying it.

So what I showed you was the high level view of the pretty pictures are getting now as a data dictionary, this is actually drilling down into what each one of these PR classes are content.  
 **Chuck Heazel**  
And this is pretty much exactly what's in the specialist specification right now.  
  
So abstracts feature.

Abstract feature is the superclass of all the POI model.  
  
This is taken out of law.  
  
This is taken out of city GML and is being re-used and up here.  
  
It's got a description, futurology, Future Ideas, mandatory all features, have a future ID.  
  
Just, this is fundamental.  
  
Descriptions optional issues useful, but not required, identifier is optional.  
  
It's useful but not required.  
  
And noted the identifiers are scoped name, which means it is not just an identifier, but also the, our scope or the context within which that identifiers defined.  
  
So, that allows me to use any type of identifier as long as it's under some sort of controlled vocabularies.  
  
So I can map into that a definition for what that identify actually means, or what the rules governing that identifier might be.

Name, which is simpler, generic nine, embarrassment and texture.  
  
Data is also optional and I can give it multiple names.  
  
Abstract feature, 12 different names, They're all valid.  
  
They really don't carry all that much baggage with them.  
So as a feature of the lifespan is an extension of the abstract feature.

It has a subclass this is also are re-used out of city GML three.  
  
What this does is, it adds for daytime.  
  
Properties to the to the feature abstract feature, where it was created, or was it deleted or terminated?  
  
What is valid from?  
  
What, when does it become valid?  
  
When does it start becoming valid?  
  
So this allows a feature to a PWA to persist, even when it's no longer valid.  
  
It also allows you to go back and see when the feature was created and when it was deleted, terminated is probably a better term because it may not be deleted but it may no longer be.  
  
Oh, considered part of the dataset.

This gives us some basic management data for all the POI.

Now keep in mind this these properties describe the pure why, not the future the POI is describing.

Once again, this is where we have the stuff which is specific to a POI.

And also keep in mind this is these are properties of the POI itself.

These are not described.

They did not describe the world except for one.

They don't actually describe the resources being described earlier.

So we have contact information.  
  
Thinking about retirement, You see our responsibility is because responsibility includes includes a field which is the role of that contract.  
  
So I can have multiple instances are contractor hemorrhage and each one with a different role, which gives me a lot of flexibility, of saying, here's a creator.  
  
If you have a problem, where the da da da da da, I can do 100 different contact information is each one with a distinct role.  
  
And I can go quick refine, which contract information is applicable to the problem I want to solve.  
  
Feature of interests.  
  
We touched on this before.  
  
This is a URI to the feature in the dataset that disappear while describes has metadata, allows me to take a separate metadata file of some sort undefined.

We don't care what it is.

What it is associated with the POI through your audit.  
  
Uri URL Center has property This is to worry POI properties plugin.  
  
So the POI properties are representations of properties of feature or feature is being described by representation, I mean, a simplified, usually, consumed by a human user.  
  
Version of a more complex concept, which is in the Future Model itself in the dataset itself.  
  
This is where we use a complex class, which we'll get to in a few minutes.  
  
We have keywords, R&D keywords at a one-on-one, what five, like VCI responsibility up above.  
  
This associates, a, scope.  
  
Meaning to each keyword.  
  
So it gives me a common piece of syntax.  
  
I can use over and over and over again, a common data structure.  
So I can put in the keyword, I can tell you, which, where the keyword is defined.

I can tell you what domain, what scope that keyword exist in silicon for, basically anything that looks like a keyword.

Constraints are legal and security constraints.

Once again, I'll check it out of 1 on 1 more thought And symbology.

Clearly, we want to be able to support symbology, ...

Conjecture that goes, where the dataset we don't know what it is.

I don't think we really can define it in this standard for.

We provide a URI which will point to the symbology information, which arguably, should be associated dataset, not would appear one.

This is the key of the whole, go back up one.  
This is really the center of the whole Standard, is this class right here.  
  
There was a question, all of the, actually, they're all optional.  
The only one which is mandatory that are mandatory has to be at least one contract inflow, and there has to be at least one feature of interest.

Uh, beyond that, if you don't want any properties, that's fine.

If you don't want keywords as fine, we don't put in any constraints except for those that absolutely have to be there.

Any other questions?

Ok, Geometry is actually technically optional, because you can have a feature that does not have geometry.

And arguably, you could have a PO, why that does not have Geometry.

Um, we kind of take the philosophy that, and unless it absolutely has to be mandatory, we leave it optional.

And when you get right down to it, an awful lot of this is not really essential in every case.  
So, you could have, on the flip side, as I can have an, a valid P of Y, which is very little content on the flip side.

I'm not making you populate something, which is irrelevant to the use case you're trying to where the problem you're trying to solve.

So, it's kind of a tradeoff between flexibility versus conformance requirements.

We've gone more with the flexibility side because there's so many different ways POS are used.

So many different targeted datasets are where we felt that minimizing the requirements, the required capabilities, minimizing the mandatory elements, were medical much more usable standard, now becomes much harder to validate.

Ok, PWA Property, Property is how we map a property from the dataset of formerly feature in the dataset, into the pure Y, that is how to recreate a Most cases, there'll be a simplified representation of a more complex property, More complex attributes of a future that's being described are built on top of the.

The attribute type class, which is to find your one-on-one or chevron one-on-one or nine, I'm sorry.  
All we really do is we added value to it.

So, the attribute class from the ISO standard says we need to find a cardinality.

How many there are we need to find the value domain and we need to define the value type.

Ok, DR, Add a Value to that.

And I have everything I need to describe the OK nature of the property.

I have a name because I have given it a name on a Crater property property, and I have the actual value.  
So this looks to be sufficient.

It allows us to basically map any attribute, any property of any feature and to a POI however you want to do it.

So gotcha for POI property.  
Future of interest, future of interest is basically a placeholder for any feature and the data set.

So it allows us to have a target for the URI which points back to the feature or features of just pure warriors representing.

It has, no, it's basically, it's a, it's a placeholder.

It's a it's a proxy for an actual implement, implement a feature, so it has no attributes.

I add a date because it's an example of a.

Code C I J comes out of ISO 1115.

2 attributes wants to date time.  
That's a mandatory attribute.  
What does the day, what does the time, Very straightforward.

But it also has a COD type or should be in today's time that's a typo friction.

No, maybe not.

Ok, anyway, the date type code is A ...

Is a registered set of values for what that take time to represent.

Just registers are copied is maintained by ISO.

Are there other organizations which have actually extended this code list?

But because it's a coder, so you can do, is being managed by somebody, would you see managers, et cetera?

Code this for the OGC Standards?

And if you identify new, valid values is basically an extendable enumeration, is a set of controlled values which can be used to describe what the stake time represents.

There's a management structure which allows you to add new values to that current list if you have a new set of requirements arrive.

So essentially, that's an extendable enumeration which is a very powerful concept.

It also allows us to define CI date once, have one set of software for processing ..., what have the semantics, the meaning of CR date change.

Easily changed.

This is a pattern that is commonly used within the ISO standards.

And it's one I strongly recommend that we adopt in some way, shape, or form for pure.

Why?

Because in a tremendous flexibility, it allows us to have.

So what you're seeing so far is a high level overview of the POI model.

This is a conceptual model.

Now the question comes, how real is this?

Isn't actually implementable.

So to compliment the model conceptual model, we did a little bit of hand jamming of a or.

Any other JSON implementation?

So, what we do is we created a geo JSON schema for PWA.

Then we collected examples of existing ...

Try to encode samples using the pure Geo JSON PR POI Schema.  
We kept trying to capture all the issues and the problems that came up, and you're going through that exercise.

So this is what we did.

So, the schema for ...

Was pulled down from the Geo JSON Web site.

We removed all the geometries except for wanting to live streaming polygon.  
We added properties around to represent a PDR classes.

And then we added JSON objects from ISO one-on-one more five.  
Now, the reason we're doing the JSON objects is that there are a lot of objects.

And when I'm Romans five ARR, good candidates for re-use and a geo JSON POW schema.

So we threw them into the JSON schema, in order to make them readily accessible to implementers for re-use within their own specialization of the geo JSON schema schema.

So this JSON schema is on the GitHub repository.

Under examples, geo JSON Schema around refused it, at least remembers a swig of used, it is publicly accessible if you want to beat on it, tell us what's wrong with it.  
Feel free to do so.

So the requirements for a pure why.

This is kind of walking through the requirements from the standards, the conceptual standard, and this is addressing how they're implemented through the Geo JSON encoder.

So first of all, is the requirement that it be a R subset of ...  
Is a certain class of tissue with lifespan.  
Um, we borrowed the encodings from ...

Sure, PWA, Shelby, comply with requirements for PWC contact info.

Remember, contact info wasn't required element of pure Y so are we borrowed the CEI responsibility type from the, basically we transpose B we converted the XML occurred in 1 9, 1 3 9.

Which you see XML encoding of ...

Five and return it And it's your choice on included here.

So we have a requirement that it has to conform with.

It has to have a future of interest to URI that's easy to do, and jaison metadata, URI, easier to Jaison, ..., property.

We talked about pure white properly, I think, in the next slide.

We have keywords.

Once again, we converted the XML encoding and 9139.

Writes we converted the encoding 19139 is symbology as a URI.  
So this is the approach we took to taking the conceptual model and instantiating it as a geo JSON schema.

Ok, POI Properties should have zero more value add attributes ..., Semantics While the value attribute, OK.

We'll go through all this.

This gets a little bit more complicated because how you actually implement the attribute type.

It's very dependent upon the technology you're using.

A lot of what's called for in the attribute type is actually represented in the cement, in the ...

Syntax of the encoding.

So, OK.  
  
Poi Property, Geo JSON JSON object definition, includes B cardinality includes the data type.

So I think we've actually met the requirements for pure property in the way we've implemented properly in the Geo JSON Schema for POI.

But there's a little bit of, uh, little bit of judgement involved here, because Is not A straight A equals B, B Equals C, C equals D Translation is kind of A goes to here, B goes over there.

She goes over here, and you gotta do the mapping of how each property each requirement, is actually represented.  
And the technology, the charter technology.

Not a straightforward matter sometimes.  
So one of the things that comes out of this is rather this is actually the most effective and efficient way of capturing POI properties.

And for that, we need implementation experience, which is one reason why we're here today.  
I'm excellent.

So I'm going to wrap it up with a couple of questions for the community, largely driven by our Geo JSON experience.

Do we need to add any more standard properties to the ...?

Has a fairly minimum set of standard properties.

These are properties which are expected to be probably common across all ...

Are, the more that needs to be there, are there other ones that have to be there needs to be mandatory?

Does a conceptual model capture everything we knew?

It was designed to be very modular, very modular, very, um, adaptable, easily adapt it for different data models, different datasets.  
Is it sufficient?

Is it true adaptable, does not have enough?

Specificity?  
Specificity, is it too rigid?  
Can we actually create conform and encoding specifications from the conceptual model?

Our hand exercise suggests that we can, but some of you I suspect you've actually worked with some of the implementation specification initiatives where they generate the implementation specification from your work or shape change generating from the UML model.

How can we do that with what we have here, or do we have to make more adjustments?

So basically consider this a beta.  
  
Tell us what's wrong with it, Tell us what's right with it.

Kind of like to make sure that when it goes out for public review, it is a 99% correct standard.