1
00:00:13,000 --> 00:00:23,000
In OSCAL, it actually came as a project that our company Polaris started. Little did we know

2
00:00:22,000 --> 00:00:28,000
it would take on a life of its own and probably be the direction of all of us for a new company

3
00:00:28,000 --> 00:00:38,000
. And so this is kind of exciting. When we first began this project the first thing we were doing is, in our needs analysis, was looking for data sources

4
00:00:37,000 --> 00:00:47,000
and trying to figure out what are the tools we have, what are the standards, what you know, what can we use, and by God's grace, we did stumble across

5
00:00:47,000 --> 00:00:57,000
. Well, of course we are you 800-53 and DNSSI before so we knew where our targets were. We didn't know OSCAL at that time, and so it was really huge for us.

6
00:00:56,000 --> 00:01:06,000
And we were able to use it to build a toolset that was quite different. So, yeah, we're excited to have this opportunity

7
00:01:06,000 --> 00:01:16,000
again in 10 days we celebrate our second year of using it and learning and realizing how little we know. So if you'll bear with us as we go through, our

8
00:01:16,000 --> 00:01:26,000
objective today is just to kind of share some of the things, how OSCAL has empowered our efforts, our development toward solving what we

9
00:01:26,000 --> 00:01:34,000
believe is such a huge fundamental problem right, which is, how do we accelerate ATOs today?

10

00:01:35,000 --> 00:01:44,000
We realize, I'm going to start out with that same statement because achieving and maintaining the compliant ATO state requires a monumental level of effort

11
00:01:44,000 --> 00:01:54,000
and I think OSCAL goes a long way and we believe it, and we believe what has been, what you guys are representing, the framework will allow more

12
00:01:53,000 --> 00:02:03,000
accurate, of course, an accelerated ATO posture, particularly as we start seeing those windows of assessments reduced down

13
00:02:03,000 --> 00:02:13,000
for continuous ATO environments where threat postures you do measured faster, it's going to make a big difference. So for us, we're very excited. What we'll do is we'll touch on just

14
00:02:13,000 --> 00:02:23,000
a few areas of where we've used OSCAL and how it has helped us address some of what we think the needs are.

15
00:02:23,000 --> 00:02:33,000
One thing, we might be a little different than the typical, we really tried to look at, we were called in to solve a problem which is a

16
00:02:33,000 --> 00:02:42,000
common user using Word documents and Excel and so, we couldn't believe it. We said, how many, how many 700 to 1000 page documents

17
00:02:42,000 --> 00:02:52,000
do you have that you're pulling in and when seening how somebody was trying to copy and paste and listening to the problems that one of these large prime contractors

18
00:02:52,000 --> 00:03:02,000
IC mainly, we were surprised at the effort and so we were just in between projects we thought, well, we'll just left our rocket science project

19

00:03:02,000 --> 00:03:12,000
where we were doing sensor fusion and stuff. So well, let's try some text analytics which we had done a lot, we had done a lot of data mining and text

20
00:03:12,000 --> 00:03:21,000
analytics for cyber, for NSA, so it wasn't that far off but it was very different. But our approach is more to recognize that

21
00:03:21,000 --> 00:03:31,000
we know the tools, in order to get OSCAL, we think with the changes because we want to see it succeed in order to see it be a permanent and

22
00:03:31,000 --> 00:03:41,000
continuing capability and framework for us, for everybody, for the industry, we believe that adoption needs to not just be on tools that can

23
00:03:41,000 --> 00:03:51,000
be adopted, but it needs to start at the center, kind of where where the critical mass of people are now. And we think that that is like on the editing environment so that

24
00:03:51,000 --> 00:04:01,000
we're going to have a heavier focus on the central repository which is the editing, being able to bring data in, the attestation work that allows us to create a submission

25
00:04:00,000 --> 00:04:05,000
. So with that said, I'll just drop down the first one

26
00:04:06,000 --> 00:04:15,000
. One of the key areas we found in OSCAL was it was a critical data source for us, allowed us are standardization, attribution, and some of the basics to allow us to build

27
00:04:15,000 --> 00:04:23,000
an environment so we knew what all the components were. we knew what the builds were, we knew how to layout

28

00:04:23,000 --> 00:04:33,000
using the framework and using baselines. We were able to establish different. But we built a form generator that can take any kind

29
00:04:33,000 --> 00:04:42,000
of OSCAL in and present different forms very quickly, make them very flexible but as well as allow a collaborative editing environment

30
00:04:42,000 --> 00:04:52,000
and of course push it out for submission and output. Now at the same time, we recognize what does OSCAL give us, it gives us the opportunity

31
00:04:52,000 --> 00:05:02,000
to have data enrichment, we have other sources pushing we can provide sources of data out, and we actually we can automate the collection

32
00:05:02,000 --> 00:05:12,000
and all the artifact collection that's necessary for proper submission. So, all we're showing

33
00:05:12,000 --> 00:05:22,000
here is we use it very heavily to lay out some of the base of foundational data layers. One of the

34
00:05:21,000 --> 00:05:31,000
things we we also found is when when we are looking at ways of trying to expedite this, our objective was how do we get things in quickly and to be able to be processed

35
00:05:31,000 --> 00:05:41,000
quickly and out? So we were trying to find importing capabilities initially as well as outputing capabilities, and we had tried many

36
00:05:40,000 --> 00:05:47,000
different things, including all the Office,

37
00:05:47,000 --> 00:05:51,000
Archer, the Office, EML stuff

38

00:05:52,000 --> 00:06:02,000
, using some of the data bindings. In this case, we'll use FedRAMP for an example. The form itself

39
00:06:01,000 --> 00:06:11,000
was going to be hard pressed to have people not modify the form or use their own form and then submit it so we didn't have any confidence

40
00:06:11,000 --> 00:06:18,000
that we were going to be able to use that binding to move data so we had to build a pretty elaborate

41
00:06:19,000 --> 00:06:28,000
import tool based on some very specific parsing and OSCAL provided all the framework for us to actually identify

42
00:06:28,000 --> 00:06:38,000
what it should be, as it could be, what's necessary, what we are looking for, and it allowed us to enhance some of the natural language processing tools we have used in the past

43
00:06:38,000 --> 00:06:48,000
and gave us kind of a baseline from which we could take something, and I apologize these are labeled here on this the slide,but on the left hand side all we're showing

44
00:06:47,000 --> 00:06:57,000
is, here's an original SSP and we can import right now from Word, PDF. Of course, we can pull from CSVs or some other
structured

45
00:06:57,000 --> 00:07:06,000
data set. It was the unstructured that were a little more tedious. So, we could pull from images in fact.

46
00:07:07,000 --> 00:07:16,000
So, on left hand side of this document you'll see that's actually a Word document and then what we have here,

47
00:07:16,000 --> 00:07:26,000
so obviously then what you see is here, it's translating

48
00:07:25,000 --> 00:07:35,000
Here's an example of the form we have where we have recreated this one, the form is native to the application so we can import from existing or we can

49
00:07:35,000 --> 00:07:45,000
create. but the form tries to give you a familiar feel. In this case with FedRAMP we tried to give something familiar that has a lot of

50
00:07:45,000 --> 00:07:55,000
the same descriptions and it should have immediate familiarity. Immediately then you go right across what you've noticed is, this is on the right

51
00:07:54,000 --> 00:08:04,000
hand side, we have our output so literally we could just, this import to output process just a few minutes, we take a brand new SSP, import

52
00:08:04,000 --> 00:08:14,000
it. We're running typically about 98.5 to 98.7, kind of average accuracy

53
00:08:15,000 --> 00:08:25,000
and we have a few things that should enhanced that. But it is very quick to give you something that looks very

54
00:08:25,000 --> 00:08:34,000
similar. We couldn't have done that layout, it would have been almost impossible for us without leveraging OSCAL. None of these are using the templates, we had just

55
00:08:34,000 --> 00:08:37,000
using representations

56
00:08:37,000 --> 00:08:42,000
. Another thing that we found is

57
00:08:43,000 --> 00:08:52,000
we're trying to show here is just to the fact

58
00:08:52,000 --> 00:09:02,000
we're able to walk through it and present the same templates in the fashion and then you'll see a corresponding

59
00:09:01,000 --> 00:09:08,000
output that we have here. These outputs are what are generated from

60
00:09:09,000 --> 00:09:19,000
the two second image and the fourth here. So the two bottom lower images are

61
00:09:18,000 --> 00:09:27,000
the actual, that's a hard copy output, of course it would be quite different than the XML submission

62
00:09:27,000 --> 00:09:37,000
. OK, so let me let me go in next one here, so same kind of thing here, what we really realize is that the

63
00:09:37,000 --> 00:09:47,000
power becomes in the validating and getting on the other real time responses from, hey you've hit this response

64
00:09:47,000 --> 00:09:56,000
point and created one, but now we have another requirement and but you can have some insight, users can actually know what's going on, so our validation of course works for the

65
00:09:56,000 --> 00:10:06,000
entire pre-submitted package for SSP or it can also facilitate just the at the very end when we want to have a

66
00:10:07,000 --> 00:10:17,000
final submission with all the components, the SAR, SAP, SSP, POAM, and RAR plus the rest of for FedRAMP, all the documents.

67
00:10:17,000 --> 00:10:26,000
Oh, I'm sorry, I didn't recognize that tone. I apologize like that.

68
00:10:26,000 --> 00:10:36,000
What we're trying to say is that with the pre-validation and the transmission is what we all want, we want to accelerate that process

69
00:10:36,000 --> 00:10:45,000
from which an ATO can be to be authorized or can maintain its authorization. And we don't think this is something

70
00:10:46,000 --> 00:10:55,000
that was even remotely close anywhere and it would have been a very large effort without OSCAL. For us and for the

71
00:10:55,000 --> 00:11:05,000
program offices or for the agencies, it's really going to change the game and it's a necessary step.

72
00:11:05,000 --> 00:11:12,000
Then the kind of the last area,

73
00:11:12,000 --> 00:11:21,000
our entire application of designed on a complete restful API design. Many of the

74
00:11:21,000 --> 00:11:31,000
XML services we have were OSCAL services but we intend to furnish all of these out as just services

75
00:11:31,000 --> 00:11:35,000
for service providers for extending

76
00:11:35,000 --> 00:11:45,000
the entire submission set to other providers. So if they're middleware providers making tools for connectivity

77
00:11:45,000 --> 00:11:55,000
to provide either query, whether it's in an x-query or XPath or they want to do an XML HTTP request

78
00:11:54,000 --> 00:12:04,000

, we can we can take and received data right now inside of a RESTful API service. We're very close to publishing this one, we're waiting until the

79
00:12:05,000 --> 00:12:15,000
final on Rev 5 FedRAMP. NIST and FedRAMP are collaborating on that. We have done quite

80
00:12:14,000 --> 00:12:24,000
a bit of work and up to the point where we had to stop in December, kind of waiting for the event for the next stages of completion. So that's

81
00:12:24,000 --> 00:12:28,000
kind of where we are on the on the RESTful services

82
00:12:29,000 --> 00:12:38,000
. We recognize that without having the ability to extend the enterprise to either receive or to serve data,

83
00:12:39,000 --> 00:12:48,000
most of the solutions will end up being a single silo and our objective was to be very extensible and to present a solution I don't think we could have done without OSCAL

84
00:12:48,000 --> 00:12:53,000
. And that's really it.

85
00:12:53,000 --> 00:13:03,000
I probably spoke way too fast, didn't I?