1
00:00:06,839 --> 00:00:13,559
Connor and I work for Telos and we

2
00:00:09,559 --> 00:00:15,120
use the Xacta Suite, that's our tool, and

3
00:00:13,559 --> 00:00:17,279
it supports

4
00:00:15,120 --> 00:00:19,199
projects going through the RMF process

5
00:00:17,279 --> 00:00:20,760
and getting accreditation. Specifically

6
00:00:19,199 --> 00:00:22,619
in this topic we're going to focus on

7
00:00:20,760 --> 00:00:24,660
the FedRAMP accreditation.

8
00:00:22,619 --> 00:00:25,740
So we've been,

9
00:00:24,660 --> 00:00:27,900
if you want to hit the next slide

10
00:00:25,740 --> 00:00:31,920
Connor

11
00:00:27,900 --> 00:00:33,780
we've been working on OSCAL for a while.

12
00:00:31,920 --> 00:00:37,020
Originally we had a customer ask

13
00:00:33,780 --> 00:00:40,379

about OSCAL or doing OSCAL back in 2018

14
00:00:37,020 --> 00:00:41,940
when OSCAL first came out

15
00:00:40,379 --> 00:00:42,920
and you can fast forward all the way

16
00:00:41,940 --> 00:00:46,440
to

17
00:00:42,920 --> 00:00:49,559
2022 when AWS used our tool to submit

18
00:00:46,440 --> 00:00:50,940
the first OSCAL SSP to FedRAMP. But we

19
00:00:49,559 --> 00:00:55,140
really want to focus on the things that

20
00:00:50,940 --> 00:00:57,600
we've been doing since then. So if we

21
00:00:55,140 --> 00:01:00,300
take a look at the next slide Xacta

22
00:00:57,600 --> 00:01:02,160
already has a pre-built XDE, we call it

23
00:01:00,300 --> 00:01:04,979
an XDE schema.

24
00:01:02,160 --> 00:01:08,100
So we are able to export all of this

25
00:01:04,979 --> 00:01:10,380
data from within the tool into an XML

26
00:01:08,100 --> 00:01:13,020
schema but the challenge is how do we

27
00:01:10,380 --> 00:01:15,420
translate the information that we're

28
00:01:13,020 --> 00:01:19,020
getting from our packages that we can

29
00:01:15,420 --> 00:01:22,020
post into the same structure and schema

30
00:01:19,020 --> 00:01:23,820
that OSCAL presents. Luckily a lot of

31
00:01:22,020 --> 00:01:26,180
this information already maps together

32
00:01:23,820 --> 00:01:28,799
so sensitive information

33
00:01:26,180 --> 00:01:31,140
lines up pretty well with OSCAL, same

34
00:01:28,799 --> 00:01:35,040
with our control implementation which is

35
00:01:31,140 --> 00:01:38,820
all of our control tests. What we

36
00:01:35,040 --> 00:01:41,100
call content OSCAL calls a catalog,

37
00:01:38,820 --> 00:01:43,740
what we call control tailoring or

38
00:01:41,100 --> 00:01:45,180

adjusting your baseline, for OSCAL it's a

39
00:01:43,740 --> 00:01:48,060
profile.

40
00:01:45,180 --> 00:01:50,759
The SSP is the entire workflow inside

41
00:01:48,060 --> 00:01:54,380
the tool and where we do our testing and

42
00:01:50,759 --> 00:01:54,380
results we have

43
00:01:56,579 --> 00:02:01,380
our POA&Ms, those are recommended

44
00:01:59,340 --> 00:02:04,280
implementations or component definitions

45
00:02:01,380 --> 00:02:07,380
so we found that our

46
00:02:04,280 --> 00:02:09,720
outputs and our data is lining up to

47
00:02:07,380 --> 00:02:12,060
OSCAL requirements. The challenge is

48
00:02:09,720 --> 00:02:15,239
the structure wasn't quite there, there was

49
00:02:12,060 --> 00:02:17,099
slight deviations to the

50
00:02:15,239 --> 00:02:19,980
requirements that we needed to do inside

51
00:02:17,099 --> 00:02:22,620
our tool to make it speak OSCAL better.

52
00:02:19,980 --> 00:02:25,640
But we ended up doing is building,

53
00:02:22,620 --> 00:02:25,640
you can click next screen

54
00:02:27,300 --> 00:02:36,000
perfect, we ended up creating a template

55
00:02:32,099 --> 00:02:39,120
in our tool dedicated to OSCAL and then

56
00:02:36,000 --> 00:02:41,220
we used FedRAMP's use case on top of

57
00:02:39,120 --> 00:02:43,800
that to make sure we had all of the

58
00:02:41,220 --> 00:02:46,980
requirements that are inside the OSCAL

59
00:02:43,800 --> 00:02:49,980
core model plus the additional FedRAMP

60
00:02:46,980 --> 00:02:53,099
requirements to meet their use case to

61
00:02:49,980 --> 00:02:56,640
build out an entire workflow that hits

62
00:02:53,099 --> 00:03:00,000
from the catalog requirements all

63
00:02:56,640 --> 00:03:02,640

the way down through to the POA&M export.

64
00:03:00,000 --> 00:03:05,160
So we leveraged, just like I'm sure

65
00:03:02,640 --> 00:03:08,340
everyone else has, the OSCAL base FedRAMP

66
00:03:05,160 --> 00:03:12,000
system security plan guide on top

67
00:03:08,340 --> 00:03:14,700
of the NIST OSCAL catalog

68
00:03:12,000 --> 00:03:16,920
and the FedRAMP OSCAL profile to really

69
00:03:14,700 --> 00:03:18,599
build out this template.

70
00:03:16,920 --> 00:03:20,879
And we've spent a lot of time working

71
00:03:18,599 --> 00:03:23,580
with the NIST and with FedRAMP and sort

72
00:03:20,879 --> 00:03:26,340
of tweaking and trying to understand

73
00:03:23,580 --> 00:03:28,440
what the core use cases are but then

74
00:03:26,340 --> 00:03:32,159
also how we can address some of the edge

75
00:03:28,440 --> 00:03:33,900
use cases as they come up because it's

76
00:03:32,159 --> 00:03:36,420
continually tweaking and adjusting and

77
00:03:33,900 --> 00:03:38,760
changing to make sure that we're hitting

78
00:03:36,420 --> 00:03:40,500
all of the requirements but also

79
00:03:38,760 --> 00:03:42,540
addressing some of the things we might

80
00:03:40,500 --> 00:03:45,060
not have thought about initially.

81
00:03:42,540 --> 00:03:47,519
And so what we ended up doing is

82
00:03:45,060 --> 00:03:49,379
creating a guide that

83
00:03:47,519 --> 00:03:52,440
goes through

84
00:03:49,379 --> 00:03:54,720
our entire OSCAL workflow

85
00:03:52,440 --> 00:03:58,440
and identifies all of the fields that

86
00:03:54,720 --> 00:04:01,799
are required for an MVP not just for

87
00:03:58,440 --> 00:04:04,319
an overall OSCAL export but to make sure

88
00:04:01,799 --> 00:04:06,480

that any of our users are also hitting

89
00:04:04,319 --> 00:04:07,860
the FedRAMP use cases

90
00:04:06,480 --> 00:04:10,140
as well.

91
00:04:07,860 --> 00:04:12,540
And it can be a bit challenging

92
00:04:10,140 --> 00:04:17,280
because we are having them

93
00:04:12,540 --> 00:04:19,139
use our tool to capture their data but

94
00:04:17,280 --> 00:04:21,660
they're not actually writing in an OSCAL

95
00:04:19,139 --> 00:04:23,639
format and that's important to note that

96
00:04:21,660 --> 00:04:26,520
our tool is going to output the OSCAL

97
00:04:23,639 --> 00:04:28,320
for them so one of the challenges is is

98
00:04:26,520 --> 00:04:30,840
if a

99
00:04:28,320 --> 00:04:33,860
user has not filled out specific

100
00:04:30,840 --> 00:04:38,280
fields or specific required information

101
00:04:33,860 --> 00:04:41,040
then that data may cause a schema

102
00:04:38,280 --> 00:04:42,540
validation issue or it could cause just

103
00:04:41,040 --> 00:04:45,060
a validator to fail because the

104
00:04:42,540 --> 00:04:47,280
information is missing. We developed a

105
00:04:45,060 --> 00:04:51,020
guide that walks them through generating

106
00:04:47,280 --> 00:04:51,020
OSCAL from start to finish.

107
00:04:52,500 --> 00:04:55,500
Connor did you want to go ahead to the

108
00:04:54,060 --> 00:04:59,100
next line

109
00:04:55,500 --> 00:05:01,320
Now none of this data means anything in

110
00:04:59,100 --> 00:05:04,919
the tool if we can't ingest the user's

111
00:05:01,320 --> 00:05:06,600
data as is. Now that's why Telos and AWS

112
00:05:04,919 --> 00:05:09,540
teamed up in the race to be the first

113
00:05:06,600 --> 00:05:11,400

organizations to submit an SSP. In order

114
00:05:09,540 --> 00:05:14,280
to accomplish this we first had to

115
00:05:11,400 --> 00:05:16,620
address our data migration issue.

116
00:05:14,280 --> 00:05:19,080
It was initially stored in the format of

117
00:05:16,620 --> 00:05:20,880
a Word document SSP. Now I'm sure all of

118
00:05:19,080 --> 00:05:23,940
you have seen a fully populated FedRAMP

119
00:05:20,880 --> 00:05:25,740
SSP and they are both thorough and

120
00:05:23,940 --> 00:05:28,440
substantial in length.

121
00:05:25,740 --> 00:05:30,780
So to solve this issue we took on a

122
00:05:28,440 --> 00:05:33,419
two-prong approach, the first prong was

123
00:05:30,780 --> 00:05:35,460
to employ the work of two solutions

124
00:05:33,419 --> 00:05:37,860
architects to manually copy and paste

125
00:05:35,460 --> 00:05:40,259
the contents that the SSP directly into

126
00:05:37,860 --> 00:05:42,720
our tool. It took these two solutions

127
00:05:40,259 --> 00:05:45,300
architects about two weeks of work

128
00:05:42,720 --> 00:05:47,039
totaling at 160 hours to migrate the

129
00:05:45,300 --> 00:05:48,900
first SSP.

130
00:05:47,039 --> 00:05:50,340
I know this number well because I was

131
00:05:48,900 --> 00:05:52,259
one of those solutions architects.

132
00:05:50,340 --> 00:05:54,180
Meanwhile our team developed an

133
00:05:52,259 --> 00:05:57,000
automation tool that read the data from

134
00:05:54,180 --> 00:05:58,800
the SSP without all that copy and paste

135
00:05:57,000 --> 00:06:01,500
and this bot could perform the same task

136
00:05:58,800 --> 00:06:05,340
in about 16 hours. So if you do the math

137
00:06:01,500 --> 00:06:07,740
that adds up to 144 hours saved total or

138
00:06:05,340 --> 00:06:09,240

per person 72 hours of my life I will

139
00:06:07,740 --> 00:06:11,039
never have to spend copying and pasting

140
00:06:09,240 --> 00:06:13,560
an entire SSP.

141
00:06:11,039 --> 00:06:16,139
Additionally this virtually eliminates

142
00:06:13,560 --> 00:06:19,580
the human error that comes from

143
00:06:16,139 --> 00:06:19,580
monotonous copy and paste.

144
00:06:19,919 --> 00:06:24,560
And we built out the same thing for

145
00:06:21,840 --> 00:06:24,560
POA&Ms.

146
00:06:24,600 --> 00:06:29,280
Now let's talk about how we go about

147
00:06:26,520 --> 00:06:32,639
generating the models.

148
00:06:29,280 --> 00:06:35,160
Here you can see how using the "manage

149
00:06:32,639 --> 00:06:38,039
OSCAL requirements" process step depicted

150
00:06:35,160 --> 00:06:40,979
on the right, we can completely customize

151
00:06:38,039 --> 00:06:42,900
each OSCAL SSP output for its specific

152
00:06:40,979 --> 00:06:45,720
use case.

153
00:06:42,900 --> 00:06:48,600
Without changing the vocabulary we can

154
00:06:45,720 --> 00:06:50,819
change the labels and tools, the fields

155
00:06:48,600 --> 00:06:52,020
for workflows to directly represent the

156
00:06:50,819 --> 00:06:53,340
requirements of their indicated

157
00:06:52,020 --> 00:06:55,080
framework.

158
00:06:53,340 --> 00:06:57,000
You can see the top field, which is

159
00:06:55,080 --> 00:06:59,400
highlighted in green there, that

160
00:06:57,000 --> 00:07:02,100
populates the description for the parent

161
00:06:59,400 --> 00:07:04,919
tag of control implementation. The

162
00:07:02,100 --> 00:07:07,680
profile catalog selection goes directly

163
00:07:04,919 --> 00:07:10,740

into the import profile and the

164
00:07:07,680 --> 00:07:13,139
namespace entry populates the NS field

165
00:07:10,740 --> 00:07:15,979
of all the proprietary props that appear

166
00:07:13,139 --> 00:07:15,979
throughout the SSP.

167
00:07:16,259 --> 00:07:19,560
We did something really cool with this as

168
00:07:17,940 --> 00:07:22,500
well because we wanted to make sure that

169
00:07:19,560 --> 00:07:25,860
we were use case agnostic so that

170
00:07:22,500 --> 00:07:29,460
namespace we add on anywhere where it

171
00:07:25,860 --> 00:07:32,580
wasn't previously defined by NIST

172
00:07:29,460 --> 00:07:34,500
and if the user is not going to use

173
00:07:32,580 --> 00:07:37,080
that namespace. So for our future use

174
00:07:34,500 --> 00:07:38,940
cases they're not FedRAMP, if they don't

175
00:07:37,080 --> 00:07:41,940
include that namespace we make sure that

176
00:07:38,940 --> 00:07:44,039
those entries are not included. So we

177
00:07:41,940 --> 00:07:46,979
have this flexibility built into the

178
00:07:44,039 --> 00:07:50,460
tool that we can now support our future

179
00:07:46,979 --> 00:07:54,919
use cases with this ability to toggle on

180
00:07:50,460 --> 00:07:54,919
and off the local definitions.

181
00:08:00,120 --> 00:08:05,160
So when modeling the data between XDE

182
00:08:03,419 --> 00:08:08,280
and OSCAL

183
00:08:05,160 --> 00:08:10,560
we can see directly how the fields

184
00:08:08,280 --> 00:08:13,440
correlate from within our tool into the

185
00:08:10,560 --> 00:08:15,599
OSCAL SSP. So on the top there you have

186
00:08:13,440 --> 00:08:17,400
the cloud service models, if you can see

187
00:08:15,599 --> 00:08:19,800
it generates a prop for each one of

188
00:08:17,400 --> 00:08:22,440

those selected and there's also the

189
00:08:19,800 --> 00:08:25,020
custom workflow that's introduced due to

190
00:08:22,440 --> 00:08:27,360
the requirement of markings or remarks

191
00:08:25,020 --> 00:08:29,160
when others selected so you see that

192
00:08:27,360 --> 00:08:31,560
field and the red popping up

193
00:08:29,160 --> 00:08:34,260
and the blue line shows it's

194
00:08:31,560 --> 00:08:37,260
translated directly into the SSP. The

195
00:08:34,260 --> 00:08:40,260
same goes for any private selection or I

196
00:08:37,260 --> 00:08:43,200
mean any hybrid selection pardon, of more

197
00:08:40,260 --> 00:08:45,180
than one cloud deployment model.

198
00:08:43,200 --> 00:08:47,399
And you'll see that hybrid cloud appears

199
00:08:45,180 --> 00:08:50,459
and the remarks populate right in there

200
00:08:47,399 --> 00:08:53,160
on the purple line. And then you also see

201
00:08:50,459 --> 00:08:55,740
how the digital identity level is split

202
00:08:53,160 --> 00:08:57,360
into the identity assurance level, the

203
00:08:55,740 --> 00:08:59,519
authenticator assurance level, and the

204
00:08:57,360 --> 00:09:01,800
federation assurance level, all with a

205
00:08:59,519 --> 00:09:05,220
value of one here.

206
00:09:01,800 --> 00:09:09,260
And if you also notice inside

207
00:09:05,220 --> 00:09:12,000
our output, you'll notice that the spaces

208
00:09:09,260 --> 00:09:14,459
and the punctuation or capitalization

209
00:09:12,000 --> 00:09:16,980
has been adjusted. So here we can

210
00:09:14,459 --> 00:09:20,640
showcase how the requirement is usually

211
00:09:16,980 --> 00:09:23,820
all lowercase letters and the spaces are

212
00:09:20,640 --> 00:09:27,180
exchanged for hyphens and so we've

213
00:09:23,820 --> 00:09:30,240

created that standard inside our output

214
00:09:27,180 --> 00:09:32,060
so as future use cases come up we can

215
00:09:30,240 --> 00:09:34,740
make sure that we're still outputting

216
00:09:32,060 --> 00:09:36,600
any of the additional roles that have

217
00:09:34,740 --> 00:09:38,160
been defined locally or any of the

218
00:09:36,600 --> 00:09:40,620
additional data points that are defined

219
00:09:38,160 --> 00:09:43,940
locally will auto convert into the

220
00:09:40,620 --> 00:09:43,940
expected standard format.

221
00:09:46,860 --> 00:09:51,839
And in order to modernize the workflow

222
00:09:49,440 --> 00:09:53,580
we built in redundancies and alternate

223
00:09:51,839 --> 00:09:56,040
data entry points throughout the process.

224
00:09:53,580 --> 00:09:57,420
As a standard it is recommended that the

225
00:09:56,040 --> 00:10:00,959
user works their way through the process

226
00:09:57,420 --> 00:10:02,700
steps shown on the left in order but so

227
00:10:00,959 --> 00:10:04,500
filling out the location step will

228
00:10:02,700 --> 00:10:06,180
populate the list page shown in the

229
00:10:04,500 --> 00:10:09,420
first green box,

230
00:10:06,180 --> 00:10:11,040
main location, secondary location etc. Now

231
00:10:09,420 --> 00:10:13,320
when filling out the next step in roles

232
00:10:11,040 --> 00:10:15,420
and parties, you'll be asked to set a

233
00:10:13,320 --> 00:10:17,160
location for the role and there you can

234
00:10:15,420 --> 00:10:20,279
see in the top right how the values

235
00:10:17,160 --> 00:10:22,200
appear along with other. This opens up a

236
00:10:20,279 --> 00:10:24,120
new set of fields allowing you to

237
00:10:22,200 --> 00:10:26,399
retroactively add a location that will

238
00:10:24,120 --> 00:10:28,320

be represented in the OSCAL export as

239
00:10:26,399 --> 00:10:30,360
shown on the bottom right with other

240
00:10:28,320 --> 00:10:32,640
location being the title and everything

241
00:10:30,360 --> 00:10:35,820
else just put in the address tag.

242
00:10:32,640 --> 00:10:37,260
This allows us to add it in stride as

243
00:10:35,820 --> 00:10:39,480
you're going through the workflow and

244
00:10:37,260 --> 00:10:41,399
not having to go back and put these

245
00:10:39,480 --> 00:10:43,440
things into previous steps so it's all

246
00:10:41,399 --> 00:10:45,899
built to to support those requirements

247
00:10:43,440 --> 00:10:47,240
enforced in OSCAL that everything is

248
00:10:45,899 --> 00:10:49,500
linked together.

249
00:10:47,240 --> 00:10:51,180
And we've tried to make this a

250
00:10:49,500 --> 00:10:52,980
little bit easier for the users as well

251
00:10:51,180 --> 00:10:55,019
because I mean if you have to fill

252
00:10:52,980 --> 00:10:56,940
it out two places or two times it

253
00:10:55,019 --> 00:10:59,760
becomes a copy paste exercise and we

254
00:10:56,940 --> 00:11:02,700
wanted to avoid that. So instead we allow

255
00:10:59,760 --> 00:11:04,620
them to identify and link these items

256
00:11:02,700 --> 00:11:07,560
directly and show the relationships

257
00:11:04,620 --> 00:11:09,720
without having to write in that there is

258
00:11:07,560 --> 00:11:12,420
a relationship and fill out all of that

259
00:11:09,720 --> 00:11:14,279
information redundantly.

260
00:11:12,420 --> 00:11:16,200
These are just small quality of life

261
00:11:14,279 --> 00:11:18,300
changes that we've made all across the

262
00:11:16,200 --> 00:11:19,800
process steps but thanks to the input of

263
00:11:18,300 --> 00:11:21,300

users we've built out an adaptive

264
00:11:19,800 --> 00:11:23,339
template that provides this level of

265
00:11:21,300 --> 00:11:25,640
flexibility across every section of the

266
00:11:23,339 --> 00:11:25,640
SSP

267
00:11:32,220 --> 00:11:36,620
Lacey did you want to talk about

268
00:11:33,720 --> 00:11:36,620
control implementation

269
00:11:36,660 --> 00:11:43,140
I can perfect,

270
00:11:40,079 --> 00:11:44,720
so Michaela knows we spend a lot of

271
00:11:43,140 --> 00:11:47,579
time really thinking about the control

272
00:11:44,720 --> 00:11:49,320
relationships and and what this would

273
00:11:47,579 --> 00:11:52,560
look like across all of the different

274
00:11:49,320 --> 00:11:55,459
use cases because we have

275
00:11:52,560 --> 00:11:58,860
systems that inherit,

276
00:11:55,459 --> 00:12:01,920
systems that provide, that both inherit

277
00:11:58,860 --> 00:12:04,260
and provide, in some cases systems that

278
00:12:01,920 --> 00:12:06,240
share. So we spent a lot of time

279
00:12:04,260 --> 00:12:09,300
thinking about what the bye components

280
00:12:06,240 --> 00:12:12,240
look like and how those will feed into

281
00:12:09,300 --> 00:12:13,980
the tool and then how,

282
00:12:12,240 --> 00:12:16,440
since we're doing implementation

283
00:12:13,980 --> 00:12:19,399
statements at the statement level to

284
00:12:16,440 --> 00:12:21,959
capture the individual implementations

285
00:12:19,399 --> 00:12:24,000
at that level, we needed to be able to

286
00:12:21,959 --> 00:12:28,380
feed them back up and tie them to the

287
00:12:24,000 --> 00:12:30,680
parent control. And so with this OSCAL

288
00:12:28,380 --> 00:12:33,180

release we're able to now identify

289
00:12:30,680 --> 00:12:35,040
leveraged authorizations as bye

290
00:12:33,180 --> 00:12:38,339
components that are sharing their

291
00:12:35,040 --> 00:12:40,019
implementation statements and also made

292
00:12:38,339 --> 00:12:44,279
it

293
00:12:40,019 --> 00:12:46,860
capable of identifying when our

294
00:12:44,279 --> 00:12:50,940
system does not have access to that shared

295
00:12:46,860 --> 00:12:53,339
SSP or that accreditation boundary

296
00:12:50,940 --> 00:12:55,320
project information. We're capturing in

297
00:12:53,339 --> 00:12:57,060
some just default data that says hey we

298
00:12:55,320 --> 00:12:59,279
don't have this data we don't have this

299
00:12:57,060 --> 00:13:01,380
information but we are saying that we're

300
00:12:59,279 --> 00:13:04,139
sharing it, we do have the leveraged

301
00:13:01,380 --> 00:13:06,600
authorization uuid. Now something

302
00:13:04,139 --> 00:13:10,459
unique here you'll notice well is those

303
00:13:06,600 --> 00:13:14,519
bye component uuids are set,

304
00:13:10,459 --> 00:13:17,639
capture the event itself against the

305
00:13:14,519 --> 00:13:19,560
statement ID, against the project

306
00:13:17,639 --> 00:13:21,660
property number,

307
00:13:19,560 --> 00:13:25,139
and so what we've done is generate a

308
00:13:21,660 --> 00:13:28,139
unique identifier for the specific meant

309
00:13:25,139 --> 00:13:31,500
that only changes when the statement

310
00:13:28,139 --> 00:13:33,839
itself changes. This way in the future

311
00:13:31,500 --> 00:13:38,279
when the validators are able to review

312
00:13:33,839 --> 00:13:41,579
the SSPs they can identify those uuid

313
00:13:38,279 --> 00:13:44,279

changes over time and be able to just

314
00:13:41,579 --> 00:13:46,560
check which control statements have

315
00:13:44,279 --> 00:13:49,560
adjusted. We wanted to make sure that our

316
00:13:46,560 --> 00:13:52,620
uuids can be leveraged to really see

317
00:13:49,560 --> 00:13:55,920
where the changes are happening. Now in

318
00:13:52,620 --> 00:13:57,959
some places we've got method five and in

319
00:13:55,920 --> 00:13:59,820
other places we use method four just

320
00:13:57,959 --> 00:14:02,880
because we haven't identified what would

321
00:13:59,820 --> 00:14:05,339
be a security relevant change and so as

322
00:14:02,880 --> 00:14:07,079
we start to identify those, ours will

323
00:14:05,339 --> 00:14:10,200
also change to method five in other

324
00:14:07,079 --> 00:14:14,959
locations so that you can lock in that

325
00:14:10,200 --> 00:14:17,459
uuid and then share that across other

326
00:14:14,959 --> 00:14:20,040
GRC tools or other tools that are going

327
00:14:17,459 --> 00:14:22,639
to be using the same generation

328
00:14:20,040 --> 00:14:22,639
method.

329
00:14:28,860 --> 00:14:34,700
And then here is our component in

330
00:14:31,380 --> 00:14:34,700
inventory relationships

331
00:14:35,639 --> 00:14:40,860
and

332
00:14:38,399 --> 00:14:42,899
just like when we were looking at how

333
00:14:40,860 --> 00:14:47,519
things tie together this is the same

334
00:14:42,899 --> 00:14:49,740
story where we've got our components,

335
00:14:47,519 --> 00:14:52,800
like our software and our operating

336
00:14:49,740 --> 00:14:55,440
systems and how they're installed in the

337
00:14:52,800 --> 00:14:58,800
inventory itself.

338
00:14:55,440 --> 00:15:00,420

You can see directly

339
00:14:58,800 --> 00:15:02,760
how they're representing the application

340
00:15:00,420 --> 00:15:04,639
on the top right, you have the software

341
00:15:02,760 --> 00:15:07,079
and whether it's installed in checkbox,

342
00:15:04,639 --> 00:15:08,940
and you can edit the software name,

343
00:15:07,079 --> 00:15:11,639
vendor version, but that's how it's shown

344
00:15:08,940 --> 00:15:12,899
in the top left with those props in the

345
00:15:11,639 --> 00:15:15,660
status state.

346
00:15:12,899 --> 00:15:18,180
And then if it's installed you can see

347
00:15:15,660 --> 00:15:20,339
that uuid and the green in the top being

348
00:15:18,180 --> 00:15:22,139
reflected under implement component in

349
00:15:20,339 --> 00:15:24,779
the bottom left,

350
00:15:22,139 --> 00:15:26,820
as well as the the host information in

351
00:15:24,779 --> 00:15:29,339
the red boxes being translated directly

352
00:15:26,820 --> 00:15:31,199
from the application into the inventory

353
00:15:29,339 --> 00:15:33,740
item that has that component implemented

354
00:15:31,199 --> 00:15:33,740
within it.

355
00:15:36,240 --> 00:15:41,699
So we've really spent a lot of time

356
00:15:37,920 --> 00:15:43,740
working on the relationship pieces and

357
00:15:41,699 --> 00:15:46,260
how all of these

358
00:15:43,740 --> 00:15:49,199
different elements within the OSCAL SSP

359
00:15:46,260 --> 00:15:52,860
tie together and should be represented

360
00:15:49,199 --> 00:15:55,560
and tied together leveraging these uuids.

361
00:15:52,860 --> 00:15:58,440
But while they're still method four we

362
00:15:55,560 --> 00:16:01,560
do want to showcase the relationship of

363
00:15:58,440 --> 00:16:03,360

what's happening inside the SSP so that

364
00:16:01,560 --> 00:16:07,160
a computer should be able to read them

365
00:16:03,360 --> 00:16:07,160
and correlate all of that data.

366
00:16:12,680 --> 00:16:18,680
This is our favorite topic, validating

367
00:16:16,139 --> 00:16:18,680
these models.

368
00:16:20,279 --> 00:16:23,699
We discussed it a little bit

369
00:16:22,019 --> 00:16:25,079
before but there are really three things

370
00:16:23,699 --> 00:16:27,180
that you want to do when you're looking

371
00:16:25,079 --> 00:16:29,820
at your OSCAL model and confirming that

372
00:16:27,180 --> 00:16:32,160
it's accurate. So you've got the schema

373
00:16:29,820 --> 00:16:34,920
validation and making sure that you are

374
00:16:32,160 --> 00:16:37,199
producing a valid OSCAL model but then

375
00:16:34,920 --> 00:16:40,380
you also have to check and confirm that

376
00:16:37,199 --> 00:16:43,620
your model represents the use case which

377
00:16:40,380 --> 00:16:46,259
is in this instance FedRAMP and FedRAMP

378
00:16:43,620 --> 00:16:48,779
has done a great job of providing a data

379
00:16:46,259 --> 00:16:51,259
validation tool, it's available online or

380
00:16:48,779 --> 00:16:53,459
you can download it depending on the

381
00:16:51,259 --> 00:16:57,180
confidentiality requirements of your

382
00:16:53,459 --> 00:17:00,360
information and test it locally.

383
00:16:57,180 --> 00:17:03,779
And then in some cases there may be

384
00:17:00,360 --> 00:17:06,179
issues with either the validator or the

385
00:17:03,779 --> 00:17:08,339
schema itself, maybe we haven't

386
00:17:06,179 --> 00:17:10,380
identified a use case or maybe there's

387
00:17:08,339 --> 00:17:12,480
an edge case that wasn't addressed

388
00:17:10,380 --> 00:17:15,360

inside the schema validator or inside

389
00:17:12,480 --> 00:17:17,939
the data validator. So not only are we

390
00:17:15,360 --> 00:17:19,620
checking our data and our structure but

391
00:17:17,939 --> 00:17:21,900
we're also having those open

392
00:17:19,620 --> 00:17:23,760
conversations with NIST and FedRAMP to

393
00:17:21,900 --> 00:17:25,740
say hey we think we might have

394
00:17:23,760 --> 00:17:27,959
identified another edge case because

395
00:17:25,740 --> 00:17:31,040
what we expected to happen is not

396
00:17:27,959 --> 00:17:31,040
happening in the tool.

397
00:17:31,500 --> 00:17:36,240
And those validator errors can be

398
00:17:34,260 --> 00:17:38,700
anything from a misalignment and

399
00:17:36,240 --> 00:17:40,919
vocabulary, whether it be privacy

400
00:17:38,700 --> 00:17:44,340
designation versus privacy sensitive was

401
00:17:40,919 --> 00:17:45,900
one misalignment and tags, or as they

402
00:17:44,340 --> 00:17:48,840
were building out these requirements

403
00:17:45,900 --> 00:17:51,720
finding things that just won't work in

404
00:17:48,840 --> 00:17:55,919
the existing use case that not every

405
00:17:51,720 --> 00:17:58,260
user on the system is going to have a

406
00:17:55,919 --> 00:18:00,660
role that they are assuming or not every

407
00:17:58,260 --> 00:18:02,640
role in this process is going to have a

408
00:18:00,660 --> 00:18:04,460
user on the system. So there's been some

409
00:18:02,640 --> 00:18:07,140
back and forth there and a lot of

410
00:18:04,460 --> 00:18:09,059
constructive meetings and just better

411
00:18:07,140 --> 00:18:10,860
shaping these requirements to support

412
00:18:09,059 --> 00:18:13,640
both sides of things.

413
00:18:10,860 --> 00:18:13,640

414
00:18:15,539 --> 00:18:21,059
So we use these proprietary validation

415
00:18:18,360 --> 00:18:23,460
tools like FedRAMP to validate the data

416
00:18:21,059 --> 00:18:25,320
input into the tool and as Lacey

417
00:18:23,460 --> 00:18:27,660
suggested earlier when talking about our

418
00:18:25,320 --> 00:18:29,580
guide, we provide guidance on how to fill

419
00:18:27,660 --> 00:18:31,980
out these fields so that they can best

420
00:18:29,580 --> 00:18:34,559
meet the framework they are attempting

421
00:18:31,980 --> 00:18:36,299
to meet. But if you don't put in the data

422
00:18:34,559 --> 00:18:38,460
you will see

423
00:18:36,299 --> 00:18:40,799
instant responses within the validator.

424
00:18:38,460 --> 00:18:43,620
So here virtual you have the option to

425
00:18:40,799 --> 00:18:45,539
say yes or no as is required but if you

426
00:18:43,620 --> 00:18:47,340
don't it'll be reflected in the output

427
00:18:45,539 --> 00:18:49,380
and

428
00:18:47,340 --> 00:18:52,200
as you can see on the right there are 16

429
00:18:49,380 --> 00:18:54,539
errors in this specific SSP for not

430
00:18:52,200 --> 00:18:56,820
specified being a value and not being an

431
00:18:54,539 --> 00:18:58,799
allowed value, it has to be yes or no. So

432
00:18:56,820 --> 00:19:00,660
we will maintain the integrity of the

433
00:18:58,799 --> 00:19:02,820
data input into the tool when exporting

434
00:19:00,660 --> 00:19:04,620
and provide guidance on how to rectify

435
00:19:02,820 --> 00:19:08,840
that so you can have an error-free

436
00:19:04,620 --> 00:19:08,840
experience when outputting your OSCAL.

437
00:19:13,500 --> 00:19:18,360
So our XDE pre-establishes these

438
00:19:16,860 --> 00:19:21,000

relationships and it's something that's

439
00:19:18,360 --> 00:19:22,799
native inside of Xacta so we didn't

440
00:19:21,000 --> 00:19:24,179
have to use our graph database or a

441
00:19:22,799 --> 00:19:26,400
Neo4j.

442
00:19:24,179 --> 00:19:29,179
We leveraged features that were

443
00:19:26,400 --> 00:19:32,460
pre-existent inside of Xacta

444
00:19:29,179 --> 00:19:34,740
and instead of using the uuids that we

445
00:19:32,460 --> 00:19:37,620
have pre-generated because our uuids

446
00:19:34,740 --> 00:19:41,880
that are native to our tool are only for

447
00:19:37,620 --> 00:19:44,220
the instance or the existence of an

448
00:19:41,880 --> 00:19:46,679
entry. But we want it to be a bit more

449
00:19:44,220 --> 00:19:49,980
specific and not just say oh it's the

450
00:19:46,679 --> 00:19:52,200
instance or existence of an entry but if

451
00:19:49,980 --> 00:19:55,400
that entry changes we wanted to be able

452
00:19:52,200 --> 00:19:55,400
to update that uuid.

453
00:19:56,240 --> 00:20:01,440
And when building out this guidance we

454
00:19:58,980 --> 00:20:03,179
also took into consideration the NIST schema

455
00:20:01,440 --> 00:20:05,700
validator and made sure that everything

456
00:20:03,179 --> 00:20:07,620
was compliant there. But there are some

457
00:20:05,700 --> 00:20:09,960
checks in the NIST schema validator that

458
00:20:07,620 --> 00:20:13,260
aren't strictly

459
00:20:09,960 --> 00:20:14,880
the schema of the data, it is also

460
00:20:13,260 --> 00:20:16,919
validating content as you can see here,

461
00:20:14,880 --> 00:20:19,980
if you input an invalid email format

462
00:20:16,919 --> 00:20:22,260
such as invalid email format it will

463
00:20:19,980 --> 00:20:23,460

show in the schema validator as well

464
00:20:22,260 --> 00:20:25,380
that

465
00:20:23,460 --> 00:20:26,820
it is not an accepted value and you need

466
00:20:25,380 --> 00:20:28,140
to go back so that's why we built the

467
00:20:26,820 --> 00:20:29,520
guidance throughout the tool so you

468
00:20:28,140 --> 00:20:30,840
don't have to wait until the end of the

469
00:20:29,520 --> 00:20:32,880
line when you're plugging it into these

470
00:20:30,840 --> 00:20:36,000
various validator tools to see that your

471
00:20:32,880 --> 00:20:39,480
data isn't up to up to scratch there.

472
00:20:36,000 --> 00:20:42,200
And something that's really helpful

473
00:20:39,480 --> 00:20:45,179
as well is after you do validate

474
00:20:42,200 --> 00:20:47,760
understanding where those errors are and

475
00:20:45,179 --> 00:20:50,160
making sure that you can reference in

476
00:20:47,760 --> 00:20:53,280
and make those changes as required so

477
00:20:50,160 --> 00:20:56,160
that you can output an additional valid

478
00:20:53,280 --> 00:20:58,880
either OSCAL or valid FedRAMP use case

479
00:20:56,160 --> 00:20:58,880
is important.

480
00:21:02,820 --> 00:21:07,039
So we've spent a lot of time talking

481
00:21:04,919 --> 00:21:09,360
about uuids

482
00:21:07,039 --> 00:21:13,080
but we do want to talk about the two

483
00:21:09,360 --> 00:21:15,440
types that we use inside of our OSCAL

484
00:21:13,080 --> 00:21:15,440
export.

485
00:21:19,200 --> 00:21:24,059
So one of them is the randomly

486
00:21:21,059 --> 00:21:26,880
generated uuid and we've done these for

487
00:21:24,059 --> 00:21:29,400
the uuids that we can create or aren't

488
00:21:26,880 --> 00:21:31,620

sure of how

489
00:21:29,400 --> 00:21:33,120
or what would be a security relevant

490
00:21:31,620 --> 00:21:35,159
change. So I think this is going to just

491
00:21:33,120 --> 00:21:37,020
be part of the continued conversation

492
00:21:35,159 --> 00:21:39,720
where we can figure out

493
00:21:37,020 --> 00:21:42,720
when those changes should change the

494
00:21:39,720 --> 00:21:45,000
uuid so that we know that a validator

495
00:21:42,720 --> 00:21:45,960
can check that against the previous

496
00:21:45,000 --> 00:21:48,480
version

497
00:21:45,960 --> 00:21:51,120
as opposed to the method five which is

498
00:21:48,480 --> 00:21:52,919
what we're leveraging for when we know

499
00:21:51,120 --> 00:21:56,700
there's a security relevant change and

500
00:21:52,919 --> 00:21:58,380
we want to update the uuid

501
00:21:56,700 --> 00:22:00,799
to say these are the ones that have

502
00:21:58,380 --> 00:22:00,799
adjusted.

503
00:22:01,500 --> 00:22:06,659
But what you will notice is as we are

504
00:22:04,200 --> 00:22:09,900
outputting the POA&M model and the

505
00:22:06,659 --> 00:22:14,039
SAP and the SAR, we are leveraging the SSP

506
00:22:09,900 --> 00:22:16,740
and the uuids in the SSP as the standard

507
00:22:14,039 --> 00:22:18,299
model for where those are referenced in

508
00:22:16,740 --> 00:22:21,240
the future.

509
00:22:18,299 --> 00:22:24,360
So if we have this system

510
00:22:21,240 --> 00:22:26,400
uuid, when that's referenced in the

511
00:22:24,360 --> 00:22:29,460
SAP in this RMF POA&M model. So you move

512
00:22:26,400 --> 00:22:32,100
the inventory items we make sure we're

513
00:22:29,460 --> 00:22:35,720

using the same uuid so it's continuous

514
00:22:32,100 --> 00:22:35,720
through all of the models.

515
00:22:45,000 --> 00:22:52,500
So Xacta already builds out

516
00:22:47,640 --> 00:22:55,440
regulations like NIST 800-53, CMMC, NIST

517
00:22:52,500 --> 00:22:58,740
171 and we currently have a library of

518
00:22:55,440 --> 00:23:01,260
roughly 250 regulations.

519
00:22:58,740 --> 00:23:04,200
We also have a script capable of

520
00:23:01,260 --> 00:23:06,780
producing OSCAL catalogs using these

521
00:23:04,200 --> 00:23:09,360
regulations and because of this Telos

522
00:23:06,780 --> 00:23:11,760
supports the expansion of OSCAL usage to

523
00:23:09,360 --> 00:23:14,700
other frameworks as well. We can do this

524
00:23:11,760 --> 00:23:16,380
because Xacta is framework agnostic. We

525
00:23:14,700 --> 00:23:17,820
we touched on this earlier and how you

526
00:23:16,380 --> 00:23:20,460
can customize your workflow, you can

527
00:23:17,820 --> 00:23:22,980
customize your output. But the only thing

528
00:23:20,460 --> 00:23:25,380
required of these control owners

529
00:23:22,980 --> 00:23:27,539
would be to verify that our catalog

530
00:23:25,380 --> 00:23:30,000
based on the regulation was an accurate

531
00:23:27,539 --> 00:23:32,460
reflection of their requirements.

532
00:23:30,000 --> 00:23:36,799
As long as the data goes into Xacta it

533
00:23:32,460 --> 00:23:36,799
can come out of that as OSCAL.

534
00:23:39,059 --> 00:23:42,179
We're hoping that this will

535
00:23:41,280 --> 00:23:43,500
help

536
00:23:42,179 --> 00:23:46,860

537
00:23:43,500 --> 00:23:48,900
bridge the usage of OSCAL because we can

538
00:23:46,860 --> 00:23:51,120

standardize those data models and

539
00:23:48,900 --> 00:23:51,960
because the data already exists in the

540
00:23:51,120 --> 00:23:54,539
tool

541
00:23:51,960 --> 00:23:57,059
we can output it and hopefully support

542
00:23:54,539 --> 00:24:00,620
OSCAL adoption across the other

543
00:23:57,059 --> 00:24:00,620
frameworks or requirements too.

544
00:24:07,500 --> 00:24:15,480
We

545
00:24:11,940 --> 00:24:15,480
have OSCAL partners

546
00:24:17,580 --> 00:24:23,600
who are using our tool to work through their

547
00:24:21,419 --> 00:24:26,880
special use cases

548
00:24:23,600 --> 00:24:29,159
where it's a challenge to

549
00:24:26,880 --> 00:24:32,159
find a solution

550
00:24:29,159 --> 00:24:35,299
and as we identify new use cases

551
00:24:32,159 --> 00:24:35,299
we've been using

552
00:25:02,820 --> 00:25:08,940
Lucy did we lose you?

553
00:25:06,720 --> 00:25:12,240

554
00:25:08,940 --> 00:25:15,320
This way we can continue

555
00:25:12,240 --> 00:25:18,419
to develop and enhance and

556
00:25:15,320 --> 00:25:20,240
progress but making sure that

557
00:25:18,419 --> 00:25:23,039
the OSCAL solutions,

558
00:25:20,240 --> 00:25:25,100
use cases, just like our

559
00:25:23,039 --> 00:25:27,419
customers from the individual system

560
00:25:25,100 --> 00:25:31,140
accreditation boundary all the way up to

561
00:25:27,419 --> 00:25:33,659
an enterprise or a cloud. So

562
00:25:31,140 --> 00:25:36,000
we continue to explore and push this

563
00:25:33,659 --> 00:25:39,480

with our customers to make sure that our

564
00:25:36,000 --> 00:25:42,500
OSCAL model and OSCAL in general fits

565
00:25:39,480 --> 00:25:42,500
their needs in their use case.

566
00:25:47,039 --> 00:25:52,919
So I can tell you as of yesterday

567
00:25:52,140 --> 00:25:56,940

568
00:25:52,919 --> 00:26:00,600
we successfully exported a POA&M model

569
00:25:56,940 --> 00:26:04,500
with only three FedRAMP validator errors

570
00:26:00,600 --> 00:26:08,100
so it's a big deal and then those issues

571
00:26:04,500 --> 00:26:10,080
seem to just be related to embedding the

572
00:26:08,100 --> 00:26:12,659
SSP. So it's something we're still trying

573
00:26:10,080 --> 00:26:14,340
to work through with FedRAMP but all of

574
00:26:12,659 --> 00:26:17,340
the information, the schema, and the

575
00:26:14,340 --> 00:26:19,799
structure checks out perfectly. So now

576
00:26:17,340 --> 00:26:22,500
that we have our POA&M model exporting,

577
00:26:19,799 --> 00:26:24,840
we're going to try to ingest the

578
00:26:22,500 --> 00:26:27,600
security test case procedures. They're

579
00:26:24,840 --> 00:26:30,059
just like we did with the SSP

580
00:26:27,600 --> 00:26:32,220
and the POA&M worksheet, we're going to

581
00:26:30,059 --> 00:26:34,799
ingest the manual versions of the

582
00:26:32,220 --> 00:26:37,740
security test procedures and translate

583
00:26:34,799 --> 00:26:40,559
that into OSCAL so we can take the

584
00:26:37,740 --> 00:26:42,960
current manual process and convert it

585
00:26:40,559 --> 00:26:44,480
into something that can be

586
00:26:42,960 --> 00:26:46,919

587
00:26:44,480 --> 00:26:49,380
OSCAL-based or OSCALized.

588
00:26:46,919 --> 00:26:52,620

After that hopefully we can output the

589
00:26:49,380 --> 00:26:57,380
aSAP and the SAR with all of the data from

590
00:26:52,620 --> 00:26:57,380
an original source into OSCAL. And

591
00:26:58,500 --> 00:27:04,200
one of our big goals is to continue

592
00:27:01,679 --> 00:27:06,360
to work with NIST and FedRAMP to get all

593
00:27:04,200 --> 00:27:09,440
of the models connected into one sort of

594
00:27:06,360 --> 00:27:09,440
deliverable package.

595
00:27:14,460 --> 00:27:17,480
Are there any questions?