1
00:00:01,199 --> 00:00:04,560
Thanks Anca
so as Anca mentioned we have these

2
00:00:04,560 --> 00:00:10,400
various OSCAL models and ... Trestle
provides a set of capabilities to help

3
00:00:10,400 --> 00:00:14,960
you edit, create and manipulate these
models and it helps you validate whether

4
00:00:14,960 --> 00:00:18,640
the format
of the generated OSCAL document is

5
00:00:18,640 --> 00:00:21,520
correct or not whether all the
constraints are satisfied as per the

6
00:00:21,520 --> 00:00:24,880
schema or not
and since these models are

7
00:00:24,880 --> 00:00:28,960
interconnected for example a profile
refers to a catalog, component definition

8
00:00:28,960 --> 00:00:34,000
refers to a catalog or profile, we also
validate based on those references

9
00:00:34,000 --> 00:00:38,719
whether the set of controls that are
getting used they are they are correct

10
00:00:38,719 --> 00:00:41,840
or not or if there are any
inconsistencies over this all those

11
00:00:41,840 --> 00:00:45,600
referential integrity constraints and
all are verified over there pressure

12
00:00:45,600 --> 00:00:48,719
also helps to
integrate with

13
00:00:48,719 --> 00:00:52,719
various third-party tools and assessment
tools

14
00:00:52,719 --> 00:00:57,680
which require to convert the data from
their native format to OSCAL and vice

15
00:00:57,680 --> 00:01:00,640
versa.
So we provide capabilities for format

16
00:01:00,640 --> 00:01:03,760
conversion you know and that's what we
are showing at the bottom where these

17
00:01:03,760 --> 00:01:07,439
OSCAL models are then pushed into
different tools they work with that and

18
00:01:07,439 --> 00:01:12,799
then the results and all gets pushed
back to the different entities

19
00:01:12,799 --> 00:01:18,000
next slide please
so uh present basically uses uh the

20
00:01:18,000 --> 00:01:23,200
defsec of patterns to
help uh build the compliance artifacts

21
00:01:23,200 --> 00:01:27,840
as code so as you know index deflector
ops all the content is managed and git

22
00:01:27,840 --> 00:01:33,040
repository s code which helps provide
visibility across different teams since

23
00:01:33,040 --> 00:01:35,920
the content different contents can be
owned by different teams they can be a

24
00:01:35,920 --> 00:01:41,040
separate catalog owner profiler ssp
owner and component definition owner

25
00:01:41,040 --> 00:01:44,720
having all the content and get and
provides the visibility to different

26
00:01:44,720 --> 00:01:50,079
teams to all this content
it also helps to automatically validate

27
00:01:50,079 --> 00:01:55,840
and verify these content whenever some
updates are pushed to this content we

28
00:01:55,840 --> 00:01:59,920
use stressful command line interface
whenever any changes happen to to

29
00:01:59,920 --> 00:02:03,840
validate whether these contents are as
per the OSCAL schema

30
00:02:03,840 --> 00:02:09,759
and also all this uh editing and updates
are logged into it so that we can have

31
00:02:09,759 --> 00:02:14,000
change tracking
and approval processes over there to

32
00:02:14,000 --> 00:02:18,400
to to so that only the approved content
gets updated and

33
00:02:18,400 --> 00:02:22,400
we keep track of who is changing
whatever content so so trestle fits very

34
00:02:22,400 --> 00:02:27,040
well in this whole def sec of pipeline
based thing uh working with git to help

35
00:02:27,040 --> 00:02:29,200
enable
the

36
00:02:29,200 --> 00:02:34,160
the manipulation of oscar documents as
code and get repositories

37
00:02:34,160 --> 00:02:39,360
next slide please so this slide
basically shows the uh high level

38
00:02:39,360 --> 00:02:45,440
architecture of tresel and its various
parts at the core we have the uh

39
00:02:45,440 --> 00:02:49,519
various oscar models converted into
programming language classes so we use

40
00:02:49,519 --> 00:02:54,000
python so all these oscar models have a
representation as python classes and

41
00:02:54,000 --> 00:02:58,959
what it helps us is it helps to validate
all the constraints on the more schema

42
00:02:58,959 --> 00:03:03,599
constraints in the in the classes itself
so whenever we are creating and editing

43
00:03:03,599 --> 00:03:07,360
it and shows that
the content is always compliant with the

44
00:03:07,360 --> 00:03:12,560
oskar schema on top of that we have an
adapter which helps us to read the oscar

45
00:03:12,560 --> 00:03:16,800
documents into into these OSCAL Python
objects and convert it back to OSCAL

46
00:03:16,800 --> 00:03:22,239
documents the base on top of this base
we have our

47
00:03:22,239 --> 00:03:26,000
set of tools to enable editing and
authoring content authoring and

48
00:03:26,000 --> 00:03:31,840
transformation so tressal basically
uses an opinionated directory structure

49
00:03:31,840 --> 00:03:36,319
where all the oscal models and contents
are stored and on top of that we have a

50
00:03:36,319 --> 00:03:41,840
set of cli command line interface to
help you import an existing oscal

51
00:03:41,840 --> 00:03:47,040
content into tresel repository
edit and create new content merge the

52
00:03:47,040 --> 00:03:51,040
content back and validate it and also so
we have command line interface for that

53
00:03:51,040 --> 00:03:55,599
and we also have an sdk so that we can
use it programmatically

54
00:03:55,599 --> 00:04:02,159
um since editing this auskal content in
json files directly is not very

55
00:04:02,159 --> 00:04:07,680
easy for humans so what pressure also
provides is a set of markdown based

56
00:04:07,680 --> 00:04:11,680
command line interface
what we allow is we convert this oscar

57
00:04:11,680 --> 00:04:17,040
documents into markdown uh documents
where users can go and edit the content

58
00:04:17,040 --> 00:04:21,120
especially the textual part of the
content where user have to provide all

59
00:04:21,120 --> 00:04:26,720
this information and then we can convert
it back to oscar so so this enables

60
00:04:26,720 --> 00:04:30,880
editing of all this content in get
repository in markdown format and then

61
00:04:30,880 --> 00:04:33,759
automatic conversion back into the OSCAL
format

62
00:04:33,759 --> 00:04:38,639
we have a set of tasks and transformers
for converting content in other formats

63
00:04:38,639 --> 00:04:42,479
like spreadsheets and tool specific
formats and all into oscar to bring this

64
00:04:42,479 --> 00:04:49,120
external content into tressel
and we also have a set of plugins to

65
00:04:49,120 --> 00:04:53,520
extend the functionality for example if
we need to do validation of fedramp sps

66
00:04:53,520 --> 00:04:56,400
and all we have built a separate uh
repository

67
00:04:56,400 --> 00:04:59,520
where we have that code and that that
can be attached to pressure using a

68
00:04:59,520 --> 00:05:02,560
plug-in mechanism
and then we have various applications

69
00:05:02,560 --> 00:05:07,600
for one application is for managing the
regulatory content and all and the

70
00:05:07,600 --> 00:05:13,520
approval workflows we have a specialized
ssp creation workflows

71
00:05:13,520 --> 00:05:18,720
in git repository and we also have
different format conversions to and from

72
00:05:18,720 --> 00:05:22,400
moscow such as the spreadsheet for
document and other native formats and

73
00:05:22,400 --> 00:05:26,160
all
next slide please

74
00:05:26,160 --> 00:05:28,479
so
this this

75
00:05:28,479 --> 00:05:32,639
shows an example of using tetracell cli
for example there is a new user who's

76
00:05:32,639 --> 00:05:36,960
not very familiar with oscar objects
they need help to manage these oscar

77
00:05:36,960 --> 00:05:42,160
files and given these OSCAL files can be
pretty huge what presell allows this to

78
00:05:42,160 --> 00:05:47,840
break these big OSCAL files into
smaller files so so here we are showing

79
00:05:47,840 --> 00:05:52,160
an example where the nosql catalog which
is having multiple groups is divided

80
00:05:52,160 --> 00:05:57,360
into separate files one per group which
users can go and edit and then we can

81
00:05:57,360 --> 00:06:02,000
merge this content back into one single
oscal file so using the split and merge

82
00:06:02,000 --> 00:06:06,639
functionality it helps easy editing of
the auscal content directly using the

83
00:06:06,639 --> 00:06:10,560
command line interface
uh next slide please

84
00:06:10,560 --> 00:06:16,639
the next requirement is around the
for the using tresellis and sdk here we

85
00:06:16,639 --> 00:06:22,479
provide a set of utilities to
transform data from a non-oscar format

86
00:06:22,479 --> 00:06:27,039
to austral and vice versa because there
are different tools they that generate

87
00:06:27,039 --> 00:06:32,000
their compliance related artifacts in
their own format and to really make

88
00:06:32,000 --> 00:06:36,960
oscar usable for them we need a set of
utilities and tools which will help them

89
00:06:36,960 --> 00:06:43,600
convert their set of a set of content
into oscar format and vice versa and

90
00:06:43,600 --> 00:06:47,680
this sdk actually forms the basis
of a

91
00:06:47,680 --> 00:06:51,199
security and compliance center in ibm
cloud where we need to convert data from

92
00:06:51,199 --> 00:06:54,639
various assessment tool formats such as
stanium and openshift compliance

93
00:06:54,639 --> 00:06:59,199
operators which generate data in their
own native format into OSCAL and so on

94
00:06:59,199 --> 00:07:03,680
and these tasks and transformers also
provide capabilities to convert other

95
00:07:03,680 --> 00:07:08,000
kind of content in different formats
such as excel sheets XML content and so

96
00:07:08,000 --> 00:07:12,160
on
into OSCAL format a couple of links are

97
00:07:12,160 --> 00:07:16,080
available here
where you can go and look at how these

98
00:07:16,080 --> 00:07:19,680
transformations can be done we have a
very detailed documentation of giving

99
00:07:19,680 --> 00:07:23,599
step by step instructions on how to use
the transformation and we also have a

100
00:07:23,599 --> 00:07:26,479
set of pre-canned demos which we have
built

101
00:07:26,479 --> 00:07:30,560
demonstrating different capabilities of
tresel so these are available

102
00:07:30,560 --> 00:07:34,560
publicly and one can go and look at each
of the demos and see how different

103
00:07:34,560 --> 00:07:40,080
capabilities of pressure can be used
next slide please.

104
00:07:40,479 --> 00:07:42,639
So

105
00:07:43,039 --> 00:07:47,680
as i mentioned right we need support for
uh authoring different kinds of content

106
00:07:47,680 --> 00:07:51,759
uh because editing oscillation is not
very

107
00:07:51,759 --> 00:07:56,479
very easy we so Trestle provides this
content authoring capability where we

108
00:07:56,479 --> 00:08:00,400
can convert content both from OSCAL
format into

109
00:08:00,400 --> 00:08:04,879
into markdown and back. And also Trestle
supports

110
00:08:04,879 --> 00:08:10,000
content authoring for even non-oscar
content for example

111

00:08:10,000 --> 00:08:13,919
a system architecture design decisions
and all which can be captured in a

112
00:08:13,919 --> 00:08:19,120
markdown format but we need to enforce
some of some templates over there and uh

113
00:08:19,120 --> 00:08:24,639
so that the content is uh
authored in a in a specified format and

114
00:08:24,639 --> 00:08:28,800
trestle can
enforce those content on that and we

115
00:08:28,800 --> 00:08:33,360
have a demo of uh
the link is given here uh on how to use

116
00:08:33,360 --> 00:08:36,880
the content authoring capability of
festival over there and we we will also

117
00:08:36,880 --> 00:08:40,880
show in
in the next few slides how we are using

118
00:08:40,880 --> 00:08:44,240
uh tesla to actually on uh author rascal
content

119
00:08:44,240 --> 00:08:48,080
next slide please
so just to add before we move forward so

120
00:08:48,080 --> 00:08:52,800
we have the set of pocs with various
personas and we

121

00:08:52,800 --> 00:08:58,399
immediately realized that depending on
the persona the set of uh

122
00:08:58,399 --> 00:09:02,160
skills right where we're
very uh

123
00:09:02,160 --> 00:09:06,320
um
on a wide spectrum right the

124
00:09:06,320 --> 00:09:10,800
uh the
farther we go from the you know policy

125
00:09:10,800 --> 00:09:13,839
as code and from for from code
developers

126
00:09:13,839 --> 00:09:18,399
uh the uh
ability to to work with uh

127
00:09:18,399 --> 00:09:23,360
raw json you know decreased and people
are more familiar with uh you know text

128
00:09:23,360 --> 00:09:28,959
or spreadsheets and so on so the what
you see next is a reflection of our you

129
00:09:28,959 --> 00:09:34,240
know a few uh
pocs with the individuals right that we

130
00:09:34,240 --> 00:09:38,240
worked with it can be that uh in you
know your

131

00:09:38,240 --> 00:09:43,120
particular context different type of
skills are required so what we present

132
00:09:43,120 --> 00:09:46,560
is by no means a limitation of what
trestle can do it is just a reflection

133
00:09:46,560 --> 00:09:52,000
of our of our pocs and if additional
type of uh

134
00:09:52,000 --> 00:09:56,880
interfaces are needed uh to for those
uh

135
00:09:56,880 --> 00:10:01,279
personas to to author that that can
threshold can be

136
00:10:01,279 --> 00:10:07,519
expanded to support those
yeah the the last piece here is the

137
00:10:07,519 --> 00:10:11,680
agile logic especially the SSP part and
since SSP

138
00:10:11,680 --> 00:10:16,079
are very complex documents and they can
be multiple editors who provide

139
00:10:16,079 --> 00:10:22,160
different kinds of contents to the ssps
what we support is we convert the ssp

140
00:10:22,160 --> 00:10:26,480
document to a markdown file per control
where different editors can go and

141

00:10:26,480 --> 00:10:30,320
provide their content in terms of
control responses and all and then these

142
00:10:30,320 --> 00:10:35,760
markdown contents can be assembled back
into a single ssp and this since we have

143
00:10:35,760 --> 00:10:40,000
this command line interface it can be
used with the ci cd pipelines or get

144
00:10:40,000 --> 00:10:44,399
workflows and also that whenever some
content changes the city pipeline

145
00:10:44,399 --> 00:10:49,040
execute the required commands to convert
this content back into austral validate

146
00:10:49,040 --> 00:10:52,720
it so that
ensure that it is following the proper

147
00:10:52,720 --> 00:10:56,560
schema and all
and then this final document ssp

148
00:10:56,560 --> 00:11:02,000
document will show we also create a word
document out of from this combined ssp

149
00:11:02,000 --> 00:11:06,880
document which can then be
given to auditors to verify whether the

150
00:11:06,880 --> 00:11:10,399
the control implementation for the
overall system is correct or not

151

00:11:10,399 --> 00:11:15,360
so next we are going to show
for each of the as i mentioned for each

152
00:11:15,360 --> 00:11:20,560
of the different users whether it is
control owners such as regulators and

153
00:11:20,560 --> 00:11:26,079
OSCAL profile owners
services owner and all how they can use

154
00:11:26,079 --> 00:11:29,760
stressel to create the content in
different formats and then convert it

155
00:11:29,760 --> 00:11:36,000
back to oscar the next slide please
so here what we are showing is

156
00:11:36,000 --> 00:11:39,760
how
a catalog can be created in the markdown

157
00:11:39,760 --> 00:11:42,959
format
and then it can be converted into

158
00:11:42,959 --> 00:11:48,959
oscillator so what we provide here is a
capability where an auscal catalog can

159
00:11:48,959 --> 00:11:52,240
be
it can be broken into separate markdown

160
00:11:52,240 --> 00:11:56,320
files one per control
users can go and edit the content over

161

00:11:56,320 --> 00:11:59,600
here in terms of they can specify the
control statement

162
00:11:59,600 --> 00:12:04,639
they can specify the guidance
even the parameters and and the default

163
00:12:04,639 --> 00:12:09,200
values and all can be specified specific
properties can also be added so there

164
00:12:09,200 --> 00:12:12,959
are two two kinds of content here one is
the standard markdown content that we

165
00:12:12,959 --> 00:12:18,240
are seeing where we specify in text
format and to specify the structured

166
00:12:18,240 --> 00:12:23,839
content like parameters and properties
we use the yaml uh format that we put it

167
00:12:23,839 --> 00:12:28,079
as a yaml header at top of the markdown
content because it's much more easier to

168
00:12:28,079 --> 00:12:31,279
capture this content in

169
00:12:34,480 --> 00:12:37,480
uh

170
00:12:41,120 --> 00:12:44,320
okay
so yes the structured content is more

171
00:12:44,320 --> 00:12:47,519
easy to capture in yammer format than

markdown

172
00:12:47,519 --> 00:12:52,240
and what we do is once all this content
is available in the markdown file we can

173
00:12:52,240 --> 00:12:56,079
easily convert that using the trestle
content authoring command line interface

174
00:12:56,079 --> 00:13:00,079
into the oscar js and so if you see on
this right hand side the text is a bit

175
00:13:00,079 --> 00:13:04,560
small sorry for that but
what we see is the parameters we create

176
00:13:04,560 --> 00:13:09,839
a control uh SC-7 here we
set the parameters and the values over

177
00:13:09,839 --> 00:13:14,160
here we set the sort id property in the
properties of that control

178
00:13:14,160 --> 00:13:19,360
the actual text of the control statement
goes into parts and statements and

179
00:13:19,360 --> 00:13:24,560
subparts of that so we create the proper
hierarchical uh format of the statement

180
00:13:24,560 --> 00:13:29,440
so if a statement has subparts like abcd
and they are further subdivided into one

181
00:13:29,440 --> 00:13:32,959
two three four and all all that

hierarchical hierarchical structure

182
00:13:32,959 --> 00:13:36,320
would be maintained in the oscar js and
when we convert this markdown into

183
00:13:36,320 --> 00:13:41,120
hospital the guidance and all would be
added as additional parts over there and

184
00:13:41,120 --> 00:13:45,040
when we when we create the oscar all the
markdown files

185
00:13:45,040 --> 00:13:48,880
for all the controls would get merged we
also have a proper directory structure

186
00:13:48,880 --> 00:13:53,279
to capture the group information so
each group would have a directory and

187
00:13:53,279 --> 00:13:57,760
then uh inside that we'll have one
markdown file for control and based on

188
00:13:57,760 --> 00:14:01,680
that directory organization when we run
the cli command this single oscal json

189
00:14:01,680 --> 00:14:05,760
file which is compliant with the
oscillator schema will be created

190
00:14:05,760 --> 00:14:10,240
the next slide please
similarly we have the capability to edit

191
00:14:10,240 --> 00:14:15,440
the profiles so here what we do is in

addition to just the content that needs

192
00:14:15,440 --> 00:14:21,120
to be edited in the profile we also
provide the content from the catalog so

193
00:14:21,120 --> 00:14:25,440
for example the control statement which
is coming is coming from the catalog

194
00:14:25,440 --> 00:14:30,480
from the catalog this would help the
profile editor to

195
00:14:30,480 --> 00:14:35,199
to look at the content so that when
the editor needs to set the values of

196
00:14:35,199 --> 00:14:40,399
the different parameters for this
control or specify additional guidance

197
00:14:40,399 --> 00:14:44,639
they can look at the control statement
so that the appropriate responses or the

198
00:14:44,639 --> 00:14:47,440
guidance can be added over there rather
than

199
00:14:47,440 --> 00:14:50,959
opening a separate window and looking at
the control statement over there all the

200
00:14:50,959 --> 00:14:56,639
content is available easily available
over here for reference and we can cut

201
00:14:56,639 --> 00:15:03,040
we can convert this uh off mark down to

oscar json format so here uh we add all

202
00:15:03,040 --> 00:15:06,720
the controls in the list of control for
this specific control the parameter

203
00:15:06,720 --> 00:15:12,639
settings would go into the modify set
parameters uh part of the profile the

204
00:15:12,639 --> 00:15:16,399
additional contents the editable content
which is added here in terms of

205
00:15:16,399 --> 00:15:21,199
additional guidance or additional parts
which would go into the control that is

206
00:15:21,199 --> 00:15:25,279
added as an alter statement over here
and the respective

207
00:15:25,279 --> 00:15:28,720
content would be added with the right
ids and properties and all and the

208
00:15:28,720 --> 00:15:33,519
content would be added over here
next slide please

209
00:15:33,519 --> 00:15:38,560
now after this we have the component
definition and there are two kinds of

210
00:15:38,560 --> 00:15:42,320
information as anchor mentioned earlier
in the component definition right one is

211
00:15:42,320 --> 00:15:48,079
the mapping from the technical rules to

the catalog controls now this kind of

212
00:15:48,079 --> 00:15:53,040
information is most easily captured in a spreadsheet rather than specifying that

213
00:15:53,040 --> 00:15:57,279
in the markdown style it can be done in magdon also but given that there will be

214
00:15:57,279 --> 00:16:01,360
lots of rules that are getting mapped to different controls

215
00:16:01,360 --> 00:16:05,519
we found at least in our experience that users are more comfortable using the

216
00:16:05,519 --> 00:16:09,040
spreadsheet and they also need to specify the corresponding parameters and

217
00:16:09,040 --> 00:16:16,240
value over here so what we do is we take the this information of mapping from

218
00:16:16,240 --> 00:16:20,399
rules to different controls for our component in the spreadsheet format and

219
00:16:20,399 --> 00:16:24,000
we convert that into an OSCAL component definition

220
00:16:24,000 --> 00:16:28,160
and again the text is not very visible over here but what we have done is we

221
00:16:28,160 --> 00:16:33,199
have created a component inside this

component definition for this vpc

222
00:16:33,199 --> 00:16:36,320
component that is shown here in the
spreadsheet.

223
00:16:36,320 --> 00:16:41,279
For that component there are three rules
which are mapping to SC-7 so what we do

224
00:16:41,279 --> 00:16:46,880
is in the implemented requirements we
have SC-7 control id under that we are

225
00:16:46,880 --> 00:16:51,600
we are capturing this rules in the form
of properties as Anca described.

226
00:16:51,600 --> 00:16:55,120
Currently
in OSCAL we don't have the support of

227
00:16:55,120 --> 00:17:00,000
capturing these technical rules and the
respected respective parameters and

228
00:17:00,000 --> 00:17:04,880
their values so we are using properties
over here to attach these rules with

229
00:17:04,880 --> 00:17:09,679
this control implementation inside the
component definition but once we have

230
00:17:09,679 --> 00:17:14,079
these capabilities to represent rules
and rule parameters properly component

231
00:17:14,079 --> 00:17:18,559
definition we will switch to the new

format once that is available

232
00:17:18,559 --> 00:17:22,559
but using the properties we are able to
associate these technical rules and

233
00:17:22,559 --> 00:17:27,679
their parameters with the the
implemented controls like sc-7 which

234
00:17:27,679 --> 00:17:31,679
which we are showing over here
in the component definition so this is

235
00:17:31,679 --> 00:17:35,200
one part of the information where we are
capturing the structured information in

236
00:17:35,200 --> 00:17:40,160
the spreadsheet and next please
the other set of information

237
00:17:40,160 --> 00:17:44,559
is the control responses in terms of the
text how a

238
00:17:44,559 --> 00:17:49,520
given service implements the given
control this information is more uh

239
00:17:49,520 --> 00:17:54,160
easily captured in the markdown format
as we have done for catalogs and profile

240
00:17:54,160 --> 00:17:58,559
so for this piece of information we
capture the control implementation

241
00:17:58,559 --> 00:18:02,160
information in this background file as

we are as we are seeing over here

242
00:18:02,160 --> 00:18:07,600
implementation a b and c where
the component definition owner can go

243
00:18:07,600 --> 00:18:12,559
and provide the control responses
we show the content from the catalogs

244
00:18:12,559 --> 00:18:17,200
and profiles and also the set of rules
that we have mapped from the spreadsheet

245
00:18:17,200 --> 00:18:21,200
into the component differentiate the top
so that the

246
00:18:21,200 --> 00:18:25,520
user who is editing this markdown for
markdown file has all the content

247
00:18:25,520 --> 00:18:30,160
available at one place to write the
appropriate control responses and then

248
00:18:30,160 --> 00:18:34,640
this gets converted into the austral
component definition format so

249
00:18:34,640 --> 00:18:38,559
we add these control responses to the
existing component definition that was

250
00:18:38,559 --> 00:18:43,200
created in the last step from
spreadsheet and it would go into the

251
00:18:43,200 --> 00:18:47,600
description of these various statement

implementations as we are seeing on the

252
00:18:47,600 --> 00:18:51,840
right hand side left
we have statement a and and there we in

253
00:18:51,840 --> 00:18:55,280
the description we have put this control
responses how this specific component

254
00:18:55,280 --> 00:19:00,000
implements this control.
Next slide please

255
00:19:00,000 --> 00:19:06,320
We also have the validation tools that
specify the mapping of technical rules

256
00:19:06,320 --> 00:19:11,600
to the checks that are implemented in
policy as code so again that information

257
00:19:11,600 --> 00:19:14,960
is captured in the spreadsheet format
and here we are showing the

258
00:19:14,960 --> 00:19:17,840
rules and the corresponding check ids
here

259
00:19:17,840 --> 00:19:21,200
in addition to the control to which this
rule belongs

260
00:19:21,200 --> 00:19:24,400
and
we can convert this excel sheet

261
00:19:24,400 --> 00:19:27,840
to an OSCAL

component definition for this validation

262
00:19:27,840 --> 00:19:31,760
component so here if we see there is a
validation component for tool chain and

263
00:19:31,760 --> 00:19:36,480
for this sc-7 control we have now two
properties one is the rule id that that

264
00:19:36,480 --> 00:19:40,559
is captured and we have the second
property which captures the check

265
00:19:40,559 --> 00:19:44,000
associated with that rule so both the
information is captured through

266
00:19:44,000 --> 00:19:48,880
properties currently and as we have
support for specifying checks and rules

267
00:19:48,880 --> 00:19:54,160
in component definition and then we'll
we will use those new constructs

268
00:19:54,160 --> 00:19:59,039
once that become available
and then next slide please so finally

269
00:19:59,039 --> 00:20:04,720
once we have all this information
just just one comment uh so now as you

270
00:20:04,720 --> 00:20:10,080
as vikas mentioned right in vpc we have
rc7 three rules and as we associate

271
00:20:10,080 --> 00:20:14,880
those three rules with the three

corresponding checks um under you know

272
00:20:14,880 --> 00:20:20,400
the the properties we use
and and their version or their you know

273
00:20:20,400 --> 00:20:24,000
the
relationship between the parameters now

274
00:20:24,000 --> 00:20:29,600
within one property entry we we
um

275
00:20:29,600 --> 00:20:33,840
with the properties we have now declared
multiple properties that we need to link

276
00:20:33,840 --> 00:20:37,440
together the right rule is the right
check the right version is the right

277
00:20:37,440 --> 00:20:41,760
version for hulu is rice version for
check so we repurpose here

278
00:20:41,760 --> 00:20:46,720
the remarks uh entry uh to provide the
key

279
00:20:46,720 --> 00:20:50,559
of the grouping right so uh because a
lot of people have asked in the past how

280
00:20:50,559 --> 00:20:55,919
how we do that and uh i know oscar is
working in the

281
00:20:55,919 --> 00:21:01,200
next release on providing support for

grouping natively so then we'll be able

282
00:21:01,200 --> 00:21:05,600
to use remarks uh properly and move to
the grouping when that would be

283
00:21:05,600 --> 00:21:09,520
available
Thanks Anca for specifying that

284
00:21:09,520 --> 00:21:13,520
yeah so so yeah we are currently using
remarks field for grouping related

285
00:21:13,520 --> 00:21:16,960
properties together and once we have the
proper construct we

286
00:21:16,960 --> 00:21:21,520
will be using that
now once we have all this content

287
00:21:21,520 --> 00:21:24,960
available the final step is to generate
the ssp

288
00:21:24,960 --> 00:21:28,400
now there are two parts here right one
is we have to

289
00:21:28,400 --> 00:21:33,520
assemble all the responses that were
added to the component destinations into

290
00:21:33,520 --> 00:21:37,039
the SSP
plus there are some additional content

291
00:21:37,039 --> 00:21:42,240
which which are specified in SSP one is

for the overall system the the system

292
00:21:42,240 --> 00:21:46,080
owner may want to specify how the
overall system implements are given

293
00:21:46,080 --> 00:21:51,039
control so that information can be added
to this markdown file in the ssp repo

294
00:21:51,039 --> 00:21:55,280
for this control and also we can capture
some of the properties like control

295
00:21:55,280 --> 00:21:59,440
origination what is the implementation
status what are the roles and all

296
00:21:59,440 --> 00:22:05,120
in the ml header and all this gets
converted into the json ssp so so as we

297
00:22:05,120 --> 00:22:08,559
are seeing on the right hand side
we have

298
00:22:08,559 --> 00:22:13,120
multiple components one per component
definition that is coming from the

299
00:22:13,120 --> 00:22:16,559
component definition OSCAL file which
gets added over here and the

300
00:22:16,559 --> 00:22:19,760
corresponding control responses are
provided

301
00:22:19,760 --> 00:22:24,480
plus we we have this system which is the

overall system for which the control

302
00:22:24,480 --> 00:22:29,360
responses are captured in this in this
markdown file and that also gets added

303
00:22:29,360 --> 00:22:33,360
as a separate component for the overall
system where this control response is

304
00:22:33,360 --> 00:22:37,600
captured over there so in this way this
ssp is almost

305
00:22:37,600 --> 00:22:40,799
automatically generated
in the sense that

306
00:22:40,799 --> 00:22:44,880
all the content is already available at
different places and we just need to

307
00:22:44,880 --> 00:22:49,120
assemble that
into this SSP there is some additional

308
00:22:49,120 --> 00:22:52,640
content that needs to be provided which
is not captured as part of the markdown

309
00:22:52,640 --> 00:22:55,360
for example the system characteristics
and all

310
00:22:55,360 --> 00:23:00,080
now these inventory information all this
information can be

311
00:23:00,080 --> 00:23:05,120
injected in into this oscar json

either if that content is available in

312
00:23:05,120 --> 00:23:09,280
markdown or JSON file in some other
places we have the capability to add

313
00:23:09,280 --> 00:23:15,200
this content into this final uh oscar
ssp that that that gets generated

314
00:23:15,200 --> 00:23:19,679
and once we have this final OSCAL json
what we do is we have capability to

315
00:23:19,679 --> 00:23:23,760
convert it
this into a word document so that

316
00:23:23,760 --> 00:23:28,880
we have all the responses with all the
uh with all the architecture

317
00:23:28,880 --> 00:23:32,640
system characteristics and everything uh
captured in the word document which can

318
00:23:32,640 --> 00:23:37,440
then be said to auditors so we have this
capability of converting oscar.json to

319
00:23:37,440 --> 00:23:42,000
what document also
already implemented yes

320
00:23:42,000 --> 00:23:44,240
going to the next slide
so

321
00:23:44,240 --> 00:23:47,919
just one more comment based on the

questions so um

322
00:23:47,919 --> 00:23:52,400
we are not forcing the people to use
markdown both the markdown and the JSON

323
00:23:52,400 --> 00:23:57,440
are available for uh for editing right
uh the the markdown is just the support

324
00:23:57,440 --> 00:24:02,240
for people that don't feel comfortable
with raw json uh but if there are

325
00:24:02,240 --> 00:24:06,480
developers in the component in the
service vendors that want to declare

326
00:24:06,480 --> 00:24:09,440
directly in the json they can go and and
uh

327
00:24:09,440 --> 00:24:14,000
directly populate the the json
yeah thanks thank you for adding so what

328
00:24:14,000 --> 00:24:17,840
we are showing here is if you see these
two arrows right whatever content we

329
00:24:17,840 --> 00:24:21,840
edit in the markdown format we can
convert back to oscar json format and if

330
00:24:21,840 --> 00:24:25,279
somebody is directly editing the json
for example this control responses are

331
00:24:25,279 --> 00:24:29,520
directly provided in the json we can

also convert that back into the markdown

332
00:24:29,520 --> 00:24:33,520
format so that users who are
more uh

333
00:24:33,520 --> 00:24:37,360
familiar with markdown and they find
json hard to edit they can go and update

334
00:24:37,360 --> 00:24:41,360
in the markdown and then we can convert
vice versa from one format to the other

335
00:24:41,360 --> 00:24:44,480
in our experience what we have
felt is

336
00:24:44,480 --> 00:24:49,039
people like uh cso team and control
owners and all they're more they're not

337
00:24:49,039 --> 00:24:53,760
very comfortable with directly editing
json so they want a format which is more

338
00:24:53,760 --> 00:24:56,799
human friendly and that's why we have
markdown but

339
00:24:56,799 --> 00:25:00,880
that doesn't stop us to stop anyone from
directly editing the json that people

340
00:25:00,880 --> 00:25:04,799
that functionality is
always available yeah

341
00:25:04,799 --> 00:25:09,679
now this uh slide basically shows the

organization of wages reports as we have

342
00:25:09,679 --> 00:25:13,840
shown we have a catalog repo where
control owners

343
00:25:13,840 --> 00:25:19,600
regulators and cso and all can go and
edit the control so we can have some

344
00:25:19,600 --> 00:25:24,880
existing catalog like this 800-53 plus
organizations can define their own

345
00:25:24,880 --> 00:25:29,200
catalogs also with additional controls
that can be created in this catalog repo

346
00:25:29,200 --> 00:25:32,880
either directly in json or in the
markdown format and converted back into

347
00:25:32,880 --> 00:25:37,120
one format and the other then we have a
separate trip of profile repo where

348
00:25:37,120 --> 00:25:41,440
additional guidance and parameters and
all can be set for for these

349
00:25:41,440 --> 00:25:43,679
controls
and

350
00:25:43,679 --> 00:25:46,720
and similarly we have the guidance repo
where the

351
00:25:46,720 --> 00:25:49,760
how the

how this control should be implemented

352
00:25:49,760 --> 00:25:54,159
by various services and all are captured
what are the evidence to be provided so

353
00:25:54,159 --> 00:25:57,440
that can be in the form of a separate
guidance report it can be part of the

354
00:25:57,440 --> 00:26:00,320
profile also we have kept these two
separate

355
00:26:00,320 --> 00:26:04,159
but this guidance support is actually an
oscar profile object only where this

356
00:26:04,159 --> 00:26:09,039
additional content gets added and this
guidance can be published in

357
00:26:09,039 --> 00:26:12,480
some website and all where users can go
and look at it so we also have a

358
00:26:12,480 --> 00:26:15,679
capability to convert that into excel
and uh

359
00:26:15,679 --> 00:26:20,880
html format over here then we have the
component definition repo where

360
00:26:20,880 --> 00:26:24,480
content can be edited in spreadsheet and
markdown format and converted back to

361
00:26:24,480 --> 00:26:28,559
the austral and then finally we have the

ssp repo where

362
00:26:28,559 --> 00:26:31,360
we
finally generate the

363
00:26:31,360 --> 00:26:37,200
SSP in the word document format
what we have done is we have linked all

364
00:26:37,200 --> 00:26:42,559
these report together in the sense that
suppose a catalog repo updates some some

365
00:26:42,559 --> 00:26:47,360
catalog or controls in this repo
what what can be done is

366
00:26:47,360 --> 00:26:52,720
once the catalog gets updated
we notify the downstream repo which are

367
00:26:52,720 --> 00:26:57,120
dependent on that catalog so in this
case we will notify the profile repo uh

368
00:26:57,120 --> 00:27:02,559
with what has changed in the catalogs
and we directly open a pr over here with

369
00:27:02,559 --> 00:27:06,559
the appropriate changes so that the
profile owner can look at the

370
00:27:06,559 --> 00:27:10,640
changes in the catalogs and then they
can merge that changes or include the

371
00:27:10,640 --> 00:27:16,559
changes in the profile repo similarly if

the profile owner updates the profile

372
00:27:16,559 --> 00:27:21,039
we notify all the downstream repo that
is the guidance component definition and

373
00:27:21,039 --> 00:27:27,120
ssp repo about this change using the git
pr mechanism so that so that whenever an

374
00:27:27,120 --> 00:27:31,440
upstream content
changes the downstream content which are

375
00:27:31,440 --> 00:27:34,960
reports which are dependent on that
content are not notified they can look

376
00:27:34,960 --> 00:27:38,399
at that content and and if they find
that reasonable then that content would

377
00:27:38,399 --> 00:27:44,320
get merged in that so this way
we have set up this end to end pipeline

378
00:27:44,320 --> 00:27:49,279
so that whenever any content changes
using the ci cd mechanism the content

379
00:27:49,279 --> 00:27:53,279
would automatically be converted to
auscal the pipelines would be triggered

380
00:27:53,279 --> 00:27:56,960
so that the next set of reports are
updated over

381
00:27:56,960 --> 00:28:01,200
are notified and whenever those reports

get updated the lower reports get

382
00:28:01,200 --> 00:28:04,320
updated
and finally the SSP can be automatically

383
00:28:04,320 --> 00:28:09,760
built using this whole uh end-to-end
pipeline across the ripples

384
00:28:10,240 --> 00:28:14,399
okay thanks thank you

385
00:28:14,399 --> 00:28:20,159
yes so uh the the spirit here is
continuous compliance right so uh the uh

386
00:28:20,159 --> 00:28:24,399
assumption is that automation is in
place in order to um

387
00:28:24,399 --> 00:28:29,679
to collect the evidence to do the
assessments and um we want to maintain

388
00:28:29,679 --> 00:28:33,679
the documentation that is associated
with that constantly up to date so that

389
00:28:33,679 --> 00:28:37,360
that's the the reason for the
implementation is pipeline and the

390
00:28:37,360 --> 00:28:41,360
uh the way that because show the
structure now um

391
00:28:41,360 --> 00:28:45,840
what we've seen so far these are um
libraries right we don't have any

392
00:28:45,840 --> 00:28:50,640
environment we may have reference
architectures that are uh uh described

393
00:28:50,640 --> 00:28:54,880
in um
like in a terraform file format or other

394
00:28:54,880 --> 00:29:01,279
other type of ci cd artifacts and we can
you know um

395
00:29:01,279 --> 00:29:08,320
associate an ssp with with that um
type of reference architecture

396
00:29:08,320 --> 00:29:13,840
artifact but now if we want to move from
this

397
00:29:14,240 --> 00:29:18,240
library context to an actual environment
context

398
00:29:18,240 --> 00:29:23,200
what we are doing is
we are looking at the

399
00:29:23,200 --> 00:29:26,640
grc
tooling right as an uh hierarchical

400
00:29:26,640 --> 00:29:30,720
architecture and i'll have a complex
live but i'll help you to traverse it

401
00:29:30,720 --> 00:29:37,360
right so we populate we expect that the
artifacts the compliances code that we

402
00:29:37,360 --> 00:29:41,440
have seen are populated for all the
controls in a particular regulation

403
00:29:41,440 --> 00:29:44,880
independent of whether they are going to
be uh

404
00:29:44,880 --> 00:29:49,440
automated or not for collecting the
posture so this information would be

405
00:29:49,440 --> 00:29:56,080
consumed from the repository by
grc in the case of ibm we have open

406
00:29:56,080 --> 00:29:59,440
pages and we developed an oscar
interface for open pages

407
00:29:59,440 --> 00:30:02,399
um
and now the um

408
00:30:02,399 --> 00:30:06,080
the this uh
depending on the scope of the various

409
00:30:06,080 --> 00:30:11,120
tools the appropriate
information will be provided so in the

410
00:30:11,120 --> 00:30:14,240
context of the
ibm

411
00:30:14,240 --> 00:30:18,480
security and compliance center because
we are a generic orchestrator all the

412
00:30:18,480 --> 00:30:23,919
technical controls details catalog
profile uh component definition will be

413
00:30:23,919 --> 00:30:29,200
populated there if you are talking about
an um an orchestrator like the openshift

414
00:30:29,200 --> 00:30:35,520
uh acm or acs we would assume that only
the controls relevant to the uh that

415
00:30:35,520 --> 00:30:40,799
particular context will be populated now
um in order to provide this information

416
00:30:40,799 --> 00:30:46,480
in a in a
standard way we have developed an um

417
00:30:46,480 --> 00:30:52,240
exchange protocol
to allow the exchange of information for

418
00:30:52,240 --> 00:30:58,000
that you know library content that we
have just seen but also uh to allow uh

419
00:30:58,000 --> 00:31:04,399
alignment to a standard for the results
for the interlock with the various tools

420
00:31:04,399 --> 00:31:08,559
that provide validation by policy
validation points so if you are looking

421
00:31:08,559 --> 00:31:13,760
at this generic orchestrator that is
working with red hat tools three ibm uh

422
00:31:13,760 --> 00:31:18,080
specialized orchestrator is tenu
cavionics they all have their native

423
00:31:18,080 --> 00:31:23,200
formats to to provide the results so we
use the exchange protocol to um

424
00:31:23,200 --> 00:31:28,720
align everything with with OSCAL the the
lower we go into this into the stack the

425
00:31:28,720 --> 00:31:32,559
more specialized we become and
eventually we hit the point where we

426
00:31:32,559 --> 00:31:38,240
have just custom interfaces but as we
are at the uh generic uh orchestrator

427
00:31:38,240 --> 00:31:44,000
level we found it very useful to to be
able to aggregate um across the uh the

428
00:31:44,000 --> 00:31:48,559
videos tools and provide posture in a
unified way so with that i will stop

429
00:31:48,559 --> 00:31:54,640
sharing and I let
Lou take over to present the details of

430
00:31:54,640 --> 00:31:58,559
the exchange protocol API