Here are four demos that I hope to share with you. The first one as Anca has explained is about the OSCAL exchange protocol that we've developed and are using for our SSP. And this is an idealized version that we've come to support based on our real life experience with the SSP and I have two links here: one is the code you can download and run yourself and try it and the second is a live demo that is also publicly available and you can go there and give it a try. This is actually kind of a work in progress. It's not complete and not fully functional but it sort of works so let's see .... So here is the GitHub repo that has the OSCAL exchange protocol code and you can go there and it explains what you minimally have to do to actually run it. You just need to have a reasonable Python and then you can just clone it and run it and so here.
here is the example that's running live that you can go to as well and and look at um and um as uh

i'll first say that this is um based on openAPI and i'm using fastAPI as the as the tool to implement this and uh as Anca mentioned, we have these uh different um life cycle operations to to communicate with the scc and um so we don't envision that each individual OSCAL document is going to be edited using this tool we envision that we're just going to deploy and manage whole documents to the scc who's going to then use those to to do their business and so what we have is actually a life cycle operation for each of the OSCAL documents that line up one for one for the different OSCAL documents and then down here we have um two kinds of validation things one validation thing is for uh policy validation points to actually get their checks for example and configure themselves to run the checks and then post
results and so we have this validation phase and we have this validation in two phases currently

we're using profiles and so we're kind of in phase one for rscc in the future um we hope
to use system security plans as the as the basis documents and so uh what uh what a pvp would do is
it would come here and say i'm this pvp please give me the set of profiles that i should be concerned about and uh there's also some filtering allowed where you can specify different uh components or other kinds of things that you want to filter so you don't get everything uh
if that's what you want to do and then when you have your results after you're running your checks you want to give them back into the the security compliance center and so we have a post that says here are my results and here's the system security plan that these results are associated with um so that's a whirlwind tour of what the uh
exchange protocol is i don't know if Anca wants to mention anything else or if i should move on
to that just to address a very important question uh on the eu ids how do we manage the year ideas so obviously if we have a full uh framework implementation right we are able to exchange uh in those um in the in the protocol the uuids together with the body of the of the information that is sent since we started um in into a phasing with a phasing approach to adopt OSCAL and the uh uh the framework uh initially we do not have uuids so when we send a profile or we send the SSP structure uh the the uuid is of course is a you know it is fake just to have a valid uh fully valid OSCAL formatted artifact but as we mature and we start now having the end-to-end support for all the elements right obviously the request for results will contain an SSP with a valid uh SSP uuid and when
the results passes back the assessment results that will contain it so this is this is a generic

00:05:13,506 --> 00:05:19,986
recommendation that we have for uh who wants to start her doctor OSCAL right start in

00:05:19,986 --> 00:05:27,106
phases and as you imagine and consider all the elements but you know you reach the maturity uh

00:05:27,106 --> 00:05:35,026
to really using them only when you have the full um the full framework uh implemented

00:05:35,666 --> 00:05:35,906
okay

00:05:38,786 --> 00:05:41,906
um so the next thing i was going to show very quickly is uh

00:05:41,906 --> 00:05:50,626
actual installation of trestle so um you could go to this link and it explains how um to do that and

00:05:51,426 --> 00:05:56,226
so this is this is it it's pretty simple to do and i'll try and do it quickly

00:06:00,626 --> 00:06:03,026
oops

00:06:24,226 --> 00:06:31,026
sorry about that

00:06:50,066 --> 00:06:50,226
all
right so i'll just show you this um so we just um
create a virtual environment and source it and
then we go ahead and say python install and
we install trestle
and then
you're able to say uh trestle um list the
commands that you're able to execute um
so um i don't know what i did much to myself in
setting up for this but i screwed myself up and i
apologize for that um and uh the next
thing i want to show really quickly
is uh the trestle markdown for uh SSP um and
this is um with Vikas and frank suits and
uh Ekaterina Nikonova from IBM Australia and
here's a link that where this demo is located
and you can navigate to this from the trestle
uh website um so let me see if i can go there
so this is the demo and as Vikas mentioned earlier you know we envisioned that different people with
different skills um are going to be uh wanting to contribute to the OSCAL process and so in um

uh in this in this uh imagined scenario um we have um in pink um OSCAL uh objects and in blue

um markdown representations and so we as uh as um Vikas described we can go back and forth between

these things um and then at the end of the day um we're gonna have uh go ahead and produce a

a markdown of an SSP and then finally a a markdown word document that would be suitable for a um

auditor to view um so um seeing that i only have two minutes left

i'm actually not going to run that demo but i'm just going to go ahead and um

quickly show the last thing i wanted to show and as again as we mentioned

um earlier uh some people are more comfortable with spreadsheets than then
using json directly and so again this is navigable from the um trestle based website

this is a demo that shows uh how a spreadsheet can be converted into a pascal document particular

component definition and so um this is what the spreadsheet would look like to the CISO

for example uh where they could have different rules and missed mappings and resources and

what comes out the other end would be a component definition um

that um represents what uh that spreadsheet uh contained and again because i have no time left

i'm not actually gonna show the demo i'm gonna give it back to Anca if she wants to wrap up

thank you oh thank you Lou this is this is very good i think uh

there are a lot of valuable um tutorials and uh demos that you can follow on the on the

website so thank you all for sharing the links and with that i'll give it back to Michaela