So today the floor is open to NIST team, to David Waltermire, OSCAL technical director and Dmitry Cousin, team member. They are going to go over some of the NIST open source tooling. So, David, I think that you are the first... right? Thank you, Michaela. I'm going to share my slides while I'm doing that.

I just wanted to say welcome everyone and thank you for the opportunity to present some of the work we've been doing here at NIST. So we've been working to see if I can close this door. So in this, we've been working on a variety of different types of tooling to support our work on the team around developing OSCAL. But we've also been keeping an eye towards building open source software that we believe will be broadly useful to the larger community. And so we've been building the software with that thinking in mind. So today I'm going to talk a little bit about some of the software that we currently have available, as well as some software that we're
currently working on for a future release.

So before I get into specific tools,

I wanted to talk a little bit about the types of tooling that we're working to provide here at NIST. Some of what we provide are what we would call programming language APIs. The intent there is to support development of, you know, business logic functionality on top of our skill-based formats as quickly as possible. So the idea is you can use a library which will give you, you know, basic parsing capabilities to marshal OSCAL content into native objects, which you can then manipulate and build higher level application on top of. So our our thoughts there is that, you know, these types of programming APIs provide an easier on ramp for developers who are trying to implement tools around scale and will ultimately support a greater amount of innovation.
We're also working on tooling that supports content conversion. So OSCAL right now has seven current models eight total planned models and point 1.1 release at the moment. And those models are available in three different formats. So at times three, that's twenty four different format model specific formats that that we that we currently maintain. We recognize that different parties are going to prefer the use of different, different formats. I may prefer YAML for editing and maybe JSON for tooling or XML. We want to make it very easy to to take a piece of OSCAL content against any model in any format and convert it to an appropriate format. for use. And this type of content conversion helps to provide a greater adaptability and flexibility within the use of OSCAL content. And then finally, we're working on content editing
capabilities, specifically browser based application that support visualizing and editing various OSCAL content. And our focus there is providing tooling and a browser that has a minimal footprint that doesn't require that you install anything or get approval to install anything. We want to make it easy for for all of you to use. So one tool that I've been working on for a few years now is, well, it's actually a library called liboscal-java. So this library is a Java API library. It provides basic code to help you parse OSCAL data and any of the available formats that allows you then to operate on that data. And in some specific ways, this is an open source project that is published out on GitHub. You can follow the link here. Access the code. So liboscal-Java has a variety of features.
One feature that it supports is what we call constraint validation. So

OSCAL as some of you might know, that OSCAL is actually modeled using a technology that we built called metaschema. It's basically a way that we can represent the structure of a model in a format agnostic way. Many schema allows us to define constraints so we can say things like if you provide this value, then you must provide this other value or we can say things like, you know, these like three fields, you must use one of the three. These types of constraint checks are not very well supported and in common schema technologies like Excel and JSON schema that we've developed. The constraint system that allows us to express a richer set of constraints on top of the OSCAL models. So this liboscal-Java tool supports enforcing those constraints on the content that it passes. It's also a multi format parser,
so it's capable of reading any of the OSCAL formats. XML, JSON and YAML

, and then writing out content ... to any of those those formats so you can read content and

... and write it back out. in and YAML

. It does that by marshalling the data to sort of a common object model that is independent of

the underlying format. And then basically writing from that independent

object model back out to the selected format

, it provides builders using

the builder pattern, which allows you to very easily construct some of the common

objects that exist within OSCAL. Things like props and links and controls and

responsible parties and

and those types of constructs. They all have builders which really simplify and make easy

generating content for these

types of objects. And it also provides a profile resolver that's currently experimental, but
it is thanks to all of the great community feedback we've been working to make it better and better. And this supports taking an OSCAL profile in any of the formats and performs the profile resolution operation on that profile. Generating resulting resolved catalog. That's all based on the latest specification that's out on the OSCAL website.

So using the liboscal-Java, we're also producing Java based command line tool called oscal-cli. So oscal-cli basically exposes all of the features that liboscal-java supports using an easy to use command line interface. It makes it easy to convert between content by loading that content and storing it in an alternate content. This allows you to write content and whatever your favorite format is, and then convert that content very easily for use in any of the other formats, such as to import into a tool or for publication.
It also allows you to quickly validate ask out content using the OSCAL constraints and resolving profiles using the profile resolver and liboscal-java. I'd like to give you just a brief demo of what this tool functions created a ... sheet here of some commands to share with you. So for this purpose, I have ... I have downloaded the OSCAL SP 800-53 Rev5 catalogue and low baseline profile, both in XML. I'll be using those to show you both the profile resolution functionality of the tool, as well as its ability to convert it to convert content into alternate formats. So First Command, I'm going to show you this is the command that you would use to convert and ask our catalog from any format into a specific format.
The oscal-cli tool actually does a deep inspection of the content to determine what its source format is. And then so you don't have to actually tell the tool what that format is. You do have to tell it what format you want to convert to. It can take to file arguments or a single argument and when to file. Arguments are provided. The first one indicates the source content. The second one, the destination file. If a single argument is provided, it reads the source content. That's specified and then writes the output, the standard out. So this is going to write a converted OSCAL catalog and the YAML format to the standard out. So it's reading the content end and then processing of that content and is writing it back out. You can see the entire OSCAL catalog now in the YAML. I can pipe this to my file if I want to.
So I can call it like catalog dot write

The next thing I wanted to show you is the ability of the tool to do profile a resolution. And one of the interesting things about this tool is you can mix, you can mix formats. So this is the command to take a profile and resolve it. The two argument indicates that you want to write the profile out, the resolved catalog out and JSON, you can see. The first argument is the profile, which is currently in XML. And the second argument is the name of the file that we want to convert.

I'll run this XML run, the profile resolver that I already have in that file. So I need to specify the overwrite command.

after to that over with the
they hold on one second.
Let me just read that file

And if that's still not right,

OK, the now it's resolving the profile and writing that file that

which it's done. So now we can show you the solved profile

in JSON format

. So that's the first few lines of the resolved profile

. And then I can take that that profile and convert it and this command will

convert

file to XML. All right, delete that

. And now it generated the XML version

of of that resolved profile. The beginning of

it converted from JSON to XML now. So this shows you like how quick and

easy this tool will allow you to basically
move between between Pascal content format

0:15:56.760,0:16:00.080
. That concludes my short demo

0:16:02.480,0:16:04.960
before I move on. You know,

0:16:04.960,0:16:11.680
this is just the start of some capabilities
that we ultimately intend to include in

0:16:12.640,0:16:20.080
and in this tool. Over time, we'd like to continue
to add features to it. One of the next things that

0:16:20.080,0:16:24.400
we'd like to start looking at
is adding functionality to

0:16:25.200,0:16:31.040
to take off scale content and
generate HTML or PDF documents

0:16:31.600,0:16:37.360
out of the oscal content
using a customizable template

0:16:38.920,0:16:48.800
. And we've got some existing XSLT based
tooling that we're currently using to do

0:16:48.800,0:16:53.520
things like generate PDFs and Excel
spreadsheets and things like that

0:16:53.520,0:16:58.320
for the eight hundred fifty three project.
One of the things that we want to work

0:16:58.320,0:17:04.720
on next is is generalizing that so that we
can integrate that into some of this tooling

0:17:04.720,0:17:07.920
for for more general use. So that'll

0:17:07.920,0:17:10.000
be a feature that should hopefully be coming in
the coming months as we continue to make progress on that.

So we also have a variety of other additional tools that I don't have time to demo today, but I'd like to touch on them a little bit.

So one area of tooling that we've invested pretty heavily in is around our continuous integration, continuous deployment environment, which we use and both the main OSCAL repo as well as in the scale content repo. And this kicked the environment, supports a variety of operations.

It allows us to take the source matter schemas for defining models and generate XML and JSON schemas generate the web based documentation that's out on the pages and instead of sites, it allows us to generate content converters, XSLT based content converters which can be used to translate content from external to
JSON or from JSON text. Now

all of this happens automatically as we as we merge pull requests into Developer Main, and this type of automation saves us a ton of time.

We also have similar functionality that exist out on the Ask Out content site, which does things like automatically convert content that we author in one format to all of the other corresponding formats. So it allows us to ontheir content and JSON, and then it will automatically generate the XML and YAML variants or if we write it in the next HTML will automatically generate the JSON and animal variants as an example. And this also saves us a bunch of time because we don't have to manually convert every file that we touch and technically, the CD is capable of being run in any repo with a little bit of setup, and
there is some instructions if you go out

0:19:37.760,0:19:42.560
at the U.S. desktop, OSCAL,
GitHub Repo and click on Build.

0:19:44.320,0:19:51.840
Another tool that we offer is
the OSCAL Deep Diff tool. So this tool

0:19:53.520,0:19:59.920
that is maintained by Nikita Wootten provides
context sensitive difference in capabilities.

0:20:01.600,0:20:04.720
So, you know, the idea
behind deep diff is that

0:20:06.160,0:20:09.680
if you're comparing a couple pieces of content

0:20:13.040,0:20:18.480
of our scale content, sometimes doing
just a simple difference over the document

0:20:18.480,0:20:19.840
is not sufficient

0:20:20.600,0:20:28.080
. Content may be moved from one section of the
document to another section of the document.

0:20:28.800,0:20:31.040
This was a problem that we

0:20:31.040,0:20:35.440
we ran into when comparing 800-

0:20:35.440,0:20:41.040
-53, rev 4 and rev5. Some
control content was merged with other controls

0:20:41.040,0:20:49.040
. Some was withdrawn. Some was created,
broken out as additional control enhancements.

0:20:49.040,0:20:51.200
And we had a really difficult time

0:20:51.200,0:20:58.400
sort of tracing the movements of content through the document. So to try to provide a better way of
understanding the changes and that type of underlying OSCCAL content, such as the 800-53 catalog, we built this deep death tool. It allows you to effectively configure how the tool does its comparison. So it allows you to do things like ignore certain elements that are relevant for comparison or identify specific fields that should be used more strongly when trying to identify if to sub portions of a document and represent the same concept. So you can do things like identify and identify our field as a key, as a key key correlate essentially, and the difference in algorithm. So if if this type of different thing is something that you're interested in, I would encourage you to go out and look at the OSCAL deep diff, the name is a little misleading. We we did build it for use with our OSCAL, but the tool itself has applicability really to any model
because of how you can configure the different

thing so it can be used with scale. But it can also be used with any other type of JSON data.

Another area that we have quite a bit of tooling around is in XSLT. If you go out to the pages that nistgov slash ... usnistgov slash or scale dash tool site, there's a variety of different XSLT demos and utilities that exist out there. A lot of them are focusing around various visualizations of different types of scale content. This is something that is being maintained by Wendell Pierce.

You'll also be able to find out there a another tool that we call oscal-cat, which we're working to to publish on that site, which Dmitry will be demoing and a quick moment. We also have a variety of programming APIs that are currently in work. So in addition to the Java API and ask our Seelye tool that I showed you earlier, we're also working on C-sharp and TypeScript now announces APIs, as well as a TypeScript notice based command line tool, as well
These are projects that are kind of in their early inception phase where, you know, building out some of the like really basic capabilities right now around, you know, loading in a model. But the plan is to continue to work on these libraries to add more and more features over time. And they're following a similar architectural style as the Java APIs is. So the hope is over time, they'll be able to provide similar kinds of features that the Java APIs currently provide. And a number of members of our team in the Nikita and Arminta are working in these areas, building out these types of APIs and tooling. Before I hand off to Dmitry, I just wanted to let you know how you can contribute to these efforts. So like all of our skill, our Open-Source tooling
is intended to be a community driven effort. So your participation and these projects can directly impact both the success of the project as well as ourselves. Larger success.

There's a few ways that you can use and contribute to these tools. You can integrate support for our scale and your tools using some of these open source projects. Your feedback on the existing tooling that we have is absolutely critical. We need to hear from you what you think is working well with the tool, what features you would like to see added. What improvements you would like to see versus how it currently works.

We also are interested in hearing from you around what other types of tooling you would like to see the US produce. We don't have a huge amount of bandwidth to build a lot of tooling, but if you have good ideas on some commodity
you know, tooling that could be used by a wide swath of the community,

it might be something that we would be interested in building. And as always, you can contribute to the development of any of these tools since they're out there as open source.

But if you have ideas on a future that you would like to add to one of these tools, you can always contribute source code and to implement that feature.

And if you're interested in coordinating with any of the devs, you can reach out to us to do that. Um,

I wanted to highlight that we also have an OSCAL tools page, which list many of these tools, as well as some community tools. The tools on that page are not are not endorsed by NIST, but as a service to the community. We are providing a listing there so that you can see what is currently out there. Um, I will reserve questions until the very end. I'll let Dmitry go next. Dmitry are you ready to hand off to you? Yes. All right. I'll stop sharing
the below and meet recursion

and I'll try to share the screen in a second. If I can find how to do it okay

Can you see my screen now?

Okay? Yes, Dmitry, we can see it. Awesome. So you know, I'll be talking and present an open security assessment language catalog authoring tool, which for short will call oscal-cat

I gave last presentation on unfinished version of the same tool on May 18th this year, and OSCAL CAT is supposed to do. It's supposed to take a catalog model of one of the instance of the catalogs. At the moment, it's only one , and build allow users to build a profile.

And further idea was that it's supposed to generate what's called profile. And then marked as TBD to resolve profiles and catalogs.

So out of the whole hierarchy of the OSCAL objects, also it operates only OSCAL
catalog model and also profile models

Currently, Cat does not have some things which would make it considerably production ready, but for professional users, it's still usable. That doesn't have some user perfection. It doesn't ask questions. Are you sure you want to do this when you can kill something critical? It doesn't appear to work with multiple catalogs since it is the work for the future, because we have to figure out how to work with multiple catalogs and in terms of profiles properly, it's not yet ready to add new controls, which didn't exist in the profile, but it has all the tooling necessary for that's basically control. There's a control, right? It's an object and typescript, so it's not a problem and doesn't have at the moment we remove .. it had a screen for back matter editing, but very few people actually want to deal with that yet. So it doesn't have all this currently not yet available features. So the tool
itself is made in ionic and angular

. If you're curious, versions are listed here.

Cat code mainly consists of TypeScript, HTML, CSS, SAS

and JSON. It is also using

QuickTime. At the moment, QuickTime was used as a library only and CSG Pipeline to remove dependency from the code deployment. But if you want local IT to generate types,

you might want to use the tool and

I saw Zach asking if there are TypeScript available entities or catalogues for all OSCAL objects like I'm using QuickTime also to generate every single entity as interfaces for TypeScript,

and it allows you to generate the most TypeScript interfaces every single object from the Schema JSON schema and then allows you to re cast the objects back in to JSON if you want. So it's quite handy. So

you were asking. So I'm recommending that there was also

a question about validation. Validation at the human level for constant dynamic
relationships is not done by this tool, either, but I'm using AGV, the tools using AGV to validate downloaded entities and of the latest version, all the entities are actually pulled out from GitHub storage of their catalogs and GitHub storage of the schemas and ... scripts. Because I'm developing on a Mac they are all based on Mac zsh, so that's basically the whole thing and a unique, angular will. Ionic is not the best tool because it allows you to compile the same source code for multiple platforms. You can make desktop application if you want using an electron compiler. You can make iOS and Android application if you feel like. But at the moment we are targeting mainly the web application and we rely upon download the availability internet, the availability of the schema, files and profile catalogs, baselines and resolved baselines by internet. But in the
case, internet connection does not work.

0:32:46.320,0:32:48.800
Application hosts that are local copies

0:32:48.800,0:32:55.840
of those files and silently falls
back to the local files in internet

0:32:55.840,0:33:01.760
upload download was not successful
on every downloaded file. Also,

0:33:02.320,0:33:08.320
application run AGV validation to make sure
that catalogs pulled out over the internet

0:33:08.320,0:33:09.840
are actually valid. Catalogs

0:33:12.400,0:33:16.480
source will become open. I'm not sure
if the repository is already open or

0:33:16.480,0:33:20.960
will become open whenever we
consider the project is ready

0:33:20.960,0:33:27.040
. It's in the it used to be in May
18th when I gave the first presentation

0:33:27.040,0:33:30.960
I used two slides. It used to be
private repository. I'm not sure because

0:33:31.680,0:33:38.720
newsstand special procedure. So a few people,
including AJ and Nikita, were helping me to

0:33:41.760,0:33:49.680
make application publication
ready. So now I'll show if you

0:33:51.680,0:33:52.180
both

0:33:56.240,0:34:02.640
few screenshots because many of the screenshots,
luckily for the commendations, I had them
So you start with the tool or with authoring mode and you have options to pick up two catalogs which are available at all GitHub region for revision five.

In this screen, you can see below the big red frame two profiles I was working on before and the goals in the title what they were derived from, which were agreed on over the council brings the authoring mode also has because it downloads the catalogs after a certain timeout, which is currently set up eight hours by default.

But they also edit settings because I was annoyed that they couldn't test this stuff unless I left the application running for eight hours to shorten the timeout. So when the catalog becomes stale, the application warns you when you select a particular type of the catalog that your you did not refresh the files from the internet
or from the local storage. So catalog baselines and profile baseline

low, moderate and high for revision four or low, moderate, high and privacy baselines also

can be refreshed using this interface

so that data information his validation for the mandatory fields

for the if you don't feel democratized on Goodreads and that's pretty much typical

life. So when you fill them in, it shows the validation. So for

the field while you're in, if you're out of the field or just turns regular

dark without special highlights. So currently the rigorous controls

, because we're in process of making drag and drop additions to the actual

are in work, but to select controls, I edit and this is new

baseline, so it has options chance of mark baselines, which

basically creates this highlights for sure control belonging to a particular baseline. So

right now we're seeing five catalog and
baselines. Actually in this particular time application, I was making screenshots from has some circle or reference, so it does not pull data properly from ... from the session. So it's highlighted everything, every single baseline present in this particular catalog type, which is a revision five. It also has an option to project baseline. So basically when you have all the catalogs present, all the controls present in a particular version of the catalog, you have option to recheck baseline and you can pick up whichever baseline you want to recheck. And after you click the appropriate baseline button basically pulls the check marks on every single control, which belongs to the baseline and also you have option to Taylor baseline, which means you cut out only controls which belong to the baseline and you end up with the effectively resolved profile. But for PreCheck Baseline, so I ordered slightly differently so well, it was a little bit my fault. So when you select, let's say, high baseline
and Project Baseline, it selects controls, which belongs to the baseline with the checkmark. So basically it's allows you to jump faster. Because I was put personally, I was annoyed for a while that I have to go and select things with the hand when I was thinking, OK, baseline is already handy anyways, so why can't I do it? And I decided to in the future. So after you select the controls, you operate on the subset of the controls you selected and I'll show you later on of the structures in the profile entity. Public Master, the particular subsections of the profile. So when you select them, you can modify controls using the modified buttons and after you're done. I didn't show retro controls because of the moment it's work in progress and after you're done with the all the tailoring you wanted to do, the particular draft of the profile,
you have choices to either save your workspace and progress and

it's allows you to make a snapshot of what you've done or metadata of the selection of the controls

so that it controls basically all the work in progress and drops. And then the session

So later on, you can start already from the place which you already worked on. But unfortunately, because it's in the application space, it's a web application. And besides store and cookies, there is not much place I can persist. Things because I like write back to the server without having actual web server

and apps or behind it. So there is also option to save the whole work space locally in the case somebody decides to clean up browser space and so on.

You can at least would be able in the future to upload the file with the current state of your editor things and hopefully resume your work further. The right Red Square highlighted is basically
allows you to save the actual profile file on your disk locally,
or you can drop it still in up space. Already compose profile so
I'll talk about the future features we're planning. Oh, and the
well making presentation and playing with the authoring mode
and statements of particular objects which are downloaded from the internet, which catalogs, profiles, baselines and so on.
I decided to add the settings screen. It's literally app settings. They're also leaving right now in the store. I tried to save them and the cookies as first books. Cookies are much more fragile than the stores and the top with the store. And they added this very scary button clear state store. Because when you begin things, sometimes they need to clean the state store and then easier to have the button for it. So I don't know. In the production, I might consider removing it or I will create some kind of level of access
and will allow it for power users,

but harder for non power users. I
don't know yet. It just then, using

as well, which I edited recently,
was scary. It worked out well

and right now in the application, I

saved the expiration interval and made it only

nine thousand of an hour, which is like a few
seconds. So original and vision features of the

was go cast were to do catalog,
which has a really simple structure

from the outside. Unless you go into the
details inside of the controls and groups of

, you should be able to feel
the profile. Very simple, right?

And then once you have profile, we were
thinking you'll use some future tools

which will allow you to pick up adjacent
objects, which OSCAL cut was the proof of

concept to show that OSCAL can be worked
on using pure JSON. But at the moment,

we don't know about any future JSON tools which
allow you to resolve profile expressed in JSON.

So it looks like we'll end
up taking JSON profile

them, convert it to XML, and using Saxon, converting it to results and the result profile.

I saw somebody's question also what is a result profile

So basically, you build profile, which tells the transformations you're applying to your original catalog and profile contains three essentially three important parts. One, that's impulse, which tells you which controls you want to pull in or which controls you want to exclude. You have Option two, well, let's say, include all and then exclude particular one. So basically, you're tailoring it. It's basically like laying the application of the papers on top of other paper, including something during something the verbs in the inside of imports input. Basically, all this notion of including something excluding something numbers it, then it has modifications. So when you modify controls, modify parameters, modifications list will contain modifications
you're doing, and then merge will describe how the modifications will be applied to

the controls, which will include also moves, the regrouping and so on. So

that was very simple. So planned features and enhancements. Once we started developing

the simple total, there was a wish list was growing. The moment they chose 18

items written on a piece of paper, but the most commonly discussed ones.

So of course, I'm currently we're going on fix bugs, which

were fine, try to make application faster or at least make it feel

faster with the synchronous loads and so on. Optimizing memory footprint

of the browser. Because sometimes I noticed that

when you work with the lot of objects and immediately pull out all four baselines

resolved baselines which are pretty big objects, catalogues and so on,

browser I have pretty beefed up computer but the browser

sometimes makes my computer sound like and from the future,
things were finding acceptable plug ins for markdown editing and rendering below editing

of the pros for the controls to make it easier. At the moment, the editing looks not pretty. Let's

put it this way for a group in controls, we're planning to add drag and drop

to make it simpler at the moment, or the lost working version had dropped down. Since it's not as user friendly as one would imagine, we'll want to a possibility to

start with an empty cat and built from scratch everything. So

basically, you start with empty object because interfaces are they are using

QuickTime. So it doesn't matter whether you start with the full catalog or with an empty one and just go. At the moment, there is no UI to add

its own UI to select. So that probably will result in development of

some other screens and UI. Plugins and resolution of the profile probably will have to go through

the path of Saxon and then JSON and XML, of the conversion
projects JSON of XML Resolved
Profile. And we add this discussing

0:45:38.240,0:45:40.240
this option of uploading.

0:45:41.600,0:45:47.560
So let's say you picked up catalog, you built
up your profile, you downloaded your profile

0:45:47.560,0:45:52.560
, but then you used external tooling or
in the future, you use our tool end to

0:45:52.560,0:45:57.520
resolve profile. So basically, you have
full catalog and then all you have right

0:45:57.520,0:46:01.440
now in UI is two original catalogs
or even five additional four. But

0:46:02.080,0:46:07.600
you have already catalog original profile,
which is a catalog you would like to use

0:46:07.600,0:46:10.240
at the moment because I'm downloading the

0:46:10.880,0:46:17.360
catalogs from online. It's not difficult
to let the user to actually upload

0:46:17.360,0:46:27.200
their own profile, result profiles, catalog
and start profile with it. It's just we have to

0:46:27.200,0:46:32.480
strictly verify the profile result profile
complies to the schema, and that's all

0:46:34.680,0:46:41.600
. So basically, all uploads of the profiles and
resolving profiles, validation and everything

0:46:41.600,0:46:43.920
should be able to done – that's another feature

0:46:44.760,0:46:48.880
. So if you have questions, we'll hold on to
the later. And right now I'll demonstrate a little bit of the application.

So I left this application running overnight so that our timeout because it now and at the moment, you can see that I selected it doesn't matter which one I select, I select, let's say Revision 4 and it prompts me right now immediately saying to refresh selected. So if I select catalog baselines low, moderate and high, so I refresh basically catalog object baselines and all the baselines together. So this checkbox basically opens the baseline. So both of this particular type of catalog when I refresh and stop scoping.

But I didn't refresh revision file. So again and I can again select everything under threshold. So I select the three and five go ahead and it tells me here that I select Revision 5 for the demo purposes I have my timer for a demo purpose I have button here, in the settings.

When I was showing you screenshots, I had the check box for the demo, I
enables also two default roles, which are defined inside of specification of 800-53, but it's not defined in the Catalog. So I add this by default and edit the few users. So I don't need to actually type by hand and say we found the presentation. So I had the responsible party, let's say font, and next to each other, you will see the responsible parties show up here and then notice for free. But still you can add and so on. I will not edit the rest of the things. So that's the longest operation, which I will say, and we're optimizing the time of the load, and I'm that probably will be forced to add some asynchronous things. So at the moment, I didn't show before, I have some this is developers branch. I have some circular reference, which is really annoying things to debug and that's great when you have several references for today's function of the
But I made that stable just for the demo so we can present. So when I say mark baselines

at the moment because session has circle of reference of marked all the baselines

which are in existence, for example, in control is not here. I think I'm running out of time. So this controls are not completely mandatory training rides for all the baseline, but they're all marked because of the way the session is a circle of reference and doesn't work properly.

Basically, all the same features as they demonstrated before it with the static snapshots have a possibility of checking baselines. I don't think it will work at the moment because, yeah, it does not because it's separate sessions back. And so for reference there, but that's what I'm debugging at the moment. I didn't do anything. So let's say I'm selecting some small. That's what I was talking about. It becomes a really, really slow.

OK, so basically, I'm selecting controls, trying to go to next
and it didn't change. It's just all right.

Basically, I'm kind of out of time, but in the end, when you select, when you edit the controls, you basically go to the download file and

That's what happens when you leave application running for eight hours in the screen. So just one last moment, the JSON result profiles are coming from historical school content dynamically. Right now, the QuickType we're using for resolving interfaces, we use schemas for validating AGV and everything is pulled in dynamically by the application. Over time, we started and also gets repository is using is the double spoke but then not hundred percent sure if it's already published from public or not, you can share looking at all. And this was an illustration to talk about profiles. I guess my time was out. Well, finally, it got to the screen.
Sorry, the application becomes really slow because I'm running from the bottom up with everything. OK, so if you have any questions, please let me know. Here is the information I'm talking. Thank you very much. And it is time for questions now.