**MARKUS KOECHL:** Good afternoon. And welcome to our session-- Best Practices, Managing Inventor iPart/iAssemblies in Vault PDM. And we really feel honored by the number of sign-ups for this class, so thank you very, very much. Of course, this is also an obligation to us, but we do our best to share a lot of how it is and best practices that we gathered together over the last few years.

My name is Markus Koechl. I started to work for Autodesk in '99, but I re-joined after a six year break in 2012. And since then, I am focused on Vault product family, especially finding solutions for any requirement our channel partners, our customers came across. And it's a pleasure to have my fellow colleague, Peter, with me for the session.

**PETER VAN AVONDT:** Yeah. Thank you Markus, for the introduction. My name is Peter van Avondt. I'm a technical specialist within Northern Europe. Specialize in the data management products or Vault, also doing a bit of fusion lifecycle. Working with Vault for 15 years, helping customers like you to improve the design processes by implementing data management strategies on a daily basis. So with this knowledge, we want to share a lot of best practices around iPart and iAssemblies managing good.

**MARKUS KOECHL:** Overall, Inventor plus Vault is a great team to manage iFamilies. So we use this artificial word to combine iParts and iAssemblies. And especially the last 4 years we did a lot of detail improvements to optimize workforce and the capabilities of the Inventor Vault add-in to deal with these table driven parts and assemblies.

Peter and myself we teamed up also three years ago, as we recognized we have common requirements to discuss. And since then, we've compiled everything we found out. And probably you already have seen in Vault Help since 2016, there is a chapter, Best Practices. So we did [INAUDIBLE] the final wording and the fine tuning, but the core content was our team effort.

In case you already are aware of this content, don't worry. For today, we have new, additional, and improved content. So this also means it's an obligation to us. After this AU we will have to update the [INAUDIBLE] file in regarding to that.

**PETER VAN AVONDT:** So within this class, we have some objectives we want to share with you. But first of all, what we really want to share this with you are this best practice we found out with working with the iAssembly/iPart within the Vault environment.

So we are going to share all the good things we've found, but also want to warn you for some pitfalls where you can run into if you're going to manage iAssemblies and iParts. We will give you a lot of tips and tricks. We'll give you read recommendations. And everything is documented in the handout that's already available, but we did some last minute updates yesterday and today even then. So we need to update it. So by the end of the week, it will be available fully updated with all recommendations in there.

So we also have a Vault backup there. It's a Vault Professional backup. But there is also the ability to download the working folder. So you have all files that we are going to use during this class, so you can run through it with the handout and see what the best practices are, OK? Good.

**MARKUS KOECHL:** We structured the content in three major chapters or blocks for today. The first one is how to set up the INVENTOR part before we start to manage in a PDM system. I'm just saying PDM system, because this probably might be applicable to any other PDM besides Vault as well.

With that, my question. Are some of you using Inventor with another non-Autodesk PDM system? OK. So in the first chapter for sure, you will get some tips as well. Block number 2, managing and walking through the entire lifecycle as well as the file handling do's and don'ts. Of course, this is really specific to Vault. It's not specific to Vault Professional. It applies to Vault Workgroup as well. I am sorry for the Vault basic users.

So setting up the table, chapter 1 will apply to this as well. But of course, run through a lifecycle, do revisions. It's Workgroup or Professional only.

Let's start with the first chapter, How to Set Up a Robust and Stable Family. Just to bring us all on the same page, where do we start? Before we jump into the first detail, we start with an existing model.

We just opened the iPart author. And before we do anything, we open the Options dialogue. And we do this because in the Options dialog, there is the first must-have. You should set must-have for our PDM purposes.

And it's all about the part number. There are two options available, both are fully supported by Vault. But you should decide for one strategy. The most common one, and this is also the default value, if you just open the Options dialogue is Set to Value. This means that you can have a start value or start letters or name, and we add just an increment. This means if your part number of the factory is 100, each member could get 100 dash 1, 2, 3, and so forth.

Of course, not all use cases and requirements are covered by this. It's just if you are allowed to have these descriptive names or numbers, and you can go with this incremental value. As I said, Inventor is actively dealing with that, and leaving with that option. It auto-fills the increment across all rows.

The second approach is don't set a value, don't preset automatically. But this also means you will have to set the value. But this value probably is driven by an ERP system or you expect probably Vault to share a file or item numbers to the table. In any case, we strictly recommend fill the table for each row. And to do this for Vault numbering schemes, today, we share also a tool of how you can fill this with just a few clicks. And this is time to hand over to Peter. We should follow this live on screen.

**PETER VAN AVONDT:** OK, so I'm going to do the hands-on today. The first example I want to show is the first option Markus talked about. So if you go to the table of an iAssembly here and edit the table-- well there's already some members in here. You will see that there is the Options over here, where you can set the part number. As you can see here, it's just examples 0, X 0. You can change this to AU 16, for example. And then at that time, it will update the part number accordingly.

This is something you have to set upfront. As Markus said before, you have to decide what you want to do. Example number one.

Example number two here is just another table, where we also predefined some members You see the part number is empty. We have the option "do not set", yeah? So what we want to do is add part numbers coming from our file numbering system from Vault.

And we had a lot of requests from customers commenting, yeah, we want to reserve a lot of numbers. I think two years ago, something like that Marcus?

**MARKUS KOECHL:** You wrote up this the idea.

**PETER VAN AVONDT:** Yeah. The idea to reserve numbers. So what we did and what we also shared on the knowledge base is a tool based on the Vault Data Standard, where you can reserve a lot of numbers. So if you really need it, install Vault Data Standard. You can easily install the reserve numbers from the attachment here.

We've got a fully documented install procedure. And what this does is adding in the tools a reserve number dialog box, giving you the ability to choose which naming scheme you want to use. We need 11 numbers reserved. At that time, it will just generate all numbers, copy to the clipboard, and we can use these numbers to just fill in the whole numbers here.

As you see, we've got a member name now, the naming for the file. You've got the member names. You can also choose another column if you really want to have the file name the same as the part number, and you can choose. But it's up to you, OK? Some questions here?

**AUDIENCE:** Is that calling from the Vault numbering scheme?

**PETER VAN AVONDT:** Yeah, this is coming from the Vault numbering scheme. So it's really reserving the file numbers out of the fold numbering schemes.

**MARKUS KOECHL:** So these are consumed in Vault, and cannot be used twice. So these are really reserved to create the members. Another question, yeah?

**AUDIENCE:** [INAUDIBLE]

**PETER VAN AVONDT:** So the question was if they can be filled out manually, or if we can add extra information on it? Yes, we can. So we can even just fill out something here, and go on, OK?

**MARKUS KOECHL:** Yeah. Our strictly recommendations is just don't leave it blank, because then Inventor will take over the [INAUDIBLE]. And consuming the member with a blank value, Inventor will fill with the following. And this is probably not the value you expect later in PDM as a part number.

**AUDIENCE:** Does the reserve-- when you reserve [INAUDIBLE].

[AUDIO OUT]

**MARKUS KOECHL:** Creates, generates the numbers. And with that, they are consumed. They are blocked in Vault. And we simply store it in the table.

**PETER VAN AVONDT:** OK. Switching back to you.

**MARKUS** There's another option in the dialogue about the variant name. To make it short, use any you

**KOECHL:** want. We do not see any issues or recommendation. It's just about usability. What your consumer, your designers prefer to get as a display name in Inventor Vault browser. But it does not impact our managing tasks later on.

Now we finished with the options, and we move forward talking about the content of the table. Of course, it's not an Inventor session. We don't talk about parameters and keys you set. I'm going to talk about iProperties, the metadata. And usually, in PDM or in data management systems, we expect some fields as mandatory to be filled, like a title. And with that, our recommendation is to add mandatory properties also in the table.

So you could use two approaches to get there. Recommendation one. Add your individual title per member already in the table. Checking in and consuming the members, Vault will read, and everything is filled, everything is done.

Approach number two. You don't add to the table, because you are saying, well, even Inventor is not able to edit iProperties, but Vault can. So I can edit Vault properties right back to the files, and everything is fine. You are right. This works.

But look at this. Reviewing the workflow to consume the members, you will see a difference in the first case. So the title values are part of the table. Regardless that you are logged in to Vault or temporarily working offline, you always create the [INAUDIBLE] member files, because these are based on the table and the metadata are part of the definition of the table.

This is different in approach number two. As long as you're logged in and consuming a member, it's always downloaded from Vault instead of recreated. So the [INAUDIBLE] Vault add-in always looks up first in Vault, does this member already exist? If yes, we grab from Vault and insert into the assembly. If not, Inventor creates if it does not already reside in Vault. So Inventor starts to create the file, and it will not add an individual value for the member we did not define before.

And so, there is no difference if the member is not pre-paired before, it's not already part of your PDM environment. It's the same effect as offline usage.

**AUDIENCE:** So in that situation, what you're saying is that the end-user who created the part is going to see the Vault [INAUDIBLE]

**MARKUS** Yeah, completely correct. Yeah. And then, hopefully, he's not able to check out the original

**KOECHL:** one residing in the Vault and overwriting with the wrongly created local one. But to avoid that, we will come back to this during the chapter two lifecycles.

**PETER VAN AVONDT:** Yep. Good. So within the dataset we are sharing, we have put two examples-- more or less the same flange, but with these two different setups on the table. So if I go to example number one, table one, you will see we've got this title column here already in the table. If you go to example number two, you will see there is no title column in the table here.

Another thing I wanted to show here is that in iProperties, the default property for the title is flange. On the other hand, the default title for the first example is set to the title of the default member. So that's a bit of difference between both of the files.

So what happens if we are going to place this into an assembly? So I'm just going to place them in Vault. And before I do, I just want to show you that in my local workspace, I emptied the members directory for flange one and flange two. So they are both empty. So if I place a member now directly from Vault-- I'm logged into Vault, I can choose one of the members. And maybe I will place a second one too, just from the first example.

You will see in the workspace, the files are downloaded. Because as you look at the date they were created depending on the time zone, you will see it was at night. I don't know. Did you work at night?

**MARKUS KOECHL:** No. We have nine hours time difference.

**PETER VAN AVONDT:** Yeah. Yeah. That's a nine hour time difference. So it just downloaded from Vault. So the same--

**AUDIENCE:** [INAUDIBLE]

**PETER VAN AVONDT:** No, because I'm not logged into Vault. Yeah, we will come back on that case later on. So I will place the second one here. So the blue one, just I think two members to do more or less the same it will be the same behavior here. And if you look at the example number two, they were created on another date just coming from Vault.

If I review, for example, the bill of materials, and we will see that the titles are just filled in correctly and fully complete as we expect to do. Now I'm going to mimic that I'm an offline user. So what I'm going to do is just log out, close Vault, and then do the same sequence.

Yeah. The only thing I need to do is place them from a local drive now. I just want to place them.

Go in here, place flange number one. And now I'm just going to add some other members here. And now we'll try to get connection to Vault. Well, in this case, I just want to mimic that I'm an offline user. I'm just going to say no. We're just going to ignore this message.

And what you will see if I go to the folder again-- just one up here-- it has created these two new flanges. Well, I'm a nine hours time zone difference, so you have to calculate the time a bit. But they are just created on the fly.

Now if I look at the bill of materials-- within the first example, it is still correct, because we added the column of this title into our table. Now if we do the same for the second example here, where we didn't add the column to the table-- just add this one. I will add the other one. You will see in the bill of material that it already adds the default value. So just to warn you, if you do this, just add this property to your table, OK?

**MARKUS KOECHL:** So for the user, probably might be worth to know that the non-existing members are downloaded from Vault, instead of created. For CAD administrators, in summary, what we discussed so far was setting the part number. Include metadata, as expected by Vault, already in the table. And for frequent consumption from a Cad manager perspective, it probably makes sense to create all members upfront at the Vault, because we saw it's more convenient for the user. They for sure get the appropriate member with all properties. And also for frequent consumption, decide for "the best" default row.

So this is the first selection inserting in Inventor. We know that's the reason that we quoted the best. It's not always easy to determine or to decide what size is the best, but try. Because if you do this later, on based on experience, it's a complete revision of the factory. And later, we will discuss impact of revisions on families.

We stay with the table contents for another aspect, and this is an option. Before we discussed mandatory field. This is clearly an option. But I hope that you will agree later on once we shared three benefits to work with this option, adding a column or an iProperty for part family.

Look at the screen shot. The part family shares one single value across all members. The intent is part number is individual for each member. Part family shares one common value across all members. And what can we do with this additional value?

Benefit number one. All members belonging to the same family, regardless which factory farm the merchant generated from, can be grouped together, either on file or item level. So this helps to increase usability and visual structure in Vault.

Benefit number two-- and probably the impact is much higher. We are flexible to drive bill of materials. Look at example one. We inserted two instances, two different sizes in our assembly. Again, it's the same component, the flange. Two different nominal diameters. And the default setting of Vault is to create items based on individual part numbers, so we get a one to one relationship. One member equals one item.

Quite often we run into the requirement that customers or companies expect to group all members to one item, but continue to differentiate on size. And with the part family option, we are able to fulfill this requirement.

Look at example number two. Again, two instances, two different sizes. But looking at first column, you will recognize we are reusing the same item. So both iPart members, both sizes are assigned to one single item. Look at the dimensions column. We continue to differentiate that each instance in the assembly is of different size.

Benefit number three. Having this common value, we easily can split our tables. Imagine you have large tables, 500, 1,000, or up to 10,000 row members. And to improve usability, or especially in performance, we can split, but with the capability to unify, either visually to group-- or even on item level, you will see that the benefit, performance increase and gain counts more.

And I think this is worth to demo how we can improve the performance. The sample file set shares a 500 member component. And you will see it takes a while until the table is read. So this could be from Inventor perspective, a reason to split the table. But we will come up with another great tip that beats everything in terms of performance.

**PETER VAN AVONDT:** Good. Just to show you a bit of set up. What we did in the assemblies is created a custom property parts family. For flexibility reasons, you can always add this column to your table here. And maybe within the same iPart, you can differentiate. Well, maybe that can be a use case, maybe not. But it can be handy to have that in the table.

So with that in place, you can use that to group your iParts, iAssemblies [INAUDIBLE] the part family. So as you can see we've got part family 0001. Here, example two file. As you can see here, an example seven is belonging to the same family and is also sorted out into this family. So that's a way of dealing with that.

So speaking of this performance, I will deal with that really shortly here. I'm just going to place one of these examples here, spacer four. It has 500 member files. If I open it up, it will take awhile to open the whole thing up. And even then, you have to sift through a lot of values.

And this example is quite not too complex, I must say. Sometimes you have a really complex iParts with a lot of rows. And so, that's not too complex here. I can place it just into my assembly.

Now, as a tip, what we can do is within Vault you can just use the search capabilities. And with Vault 2017, we have this additional inserts into CAD. So you can just select the member file directly from your members directory. Leveraging this part family, you can search on the part family, you can search on certain criteria. I did a really simple search here.

So the insert into CAD is something you can download from the Exchange Store. So it's an app that's is delivered with the 2017 release.

I'm not sure if we're going to show the item, because we have to speed up a bit. Yeah. So we will switch over. We had another demo over onto iTunes, and so on. But it's fully documented also in the PowerPoint and in the hands-on.

**MARKUS KOECHL:** With that, probably you agree that it makes sense to add to our summary list, that it's of value for the consumer having the member files created before for quick re-use and consumption, especially with a search from Vault and directly insert to CAD.

There is another aspect to finish here. We strictly recommend always create your member files setting up the family in checked out state, especially if iLogic rules are in place or if you expect accurate physical properties written to the file and also displayed in your PDF. If you don't do, then the add-in is not able to write the values to activate in the factory and create the members accordingly.

**AUDIENCE:** [INAUDIBLE]

**MARKUS KOECHL:** This would be my question back. Are you talking about Inventor library path or Vault library folder type?

**AUDIENCE:** So there's Inventor library path mapped to a Vault library folder.

**MARKUS KOECHL:** OK. No difference. Yep. Works as well.

However, especially with iAssemblies, we see you put more restrictive behavior on Inventor as needed, using this [INAUDIBLE] or protect file based library path. And in PDM environment, we think you no longer need to deal with the Inventor library path. But it's probably a different story to discuss in detail.

OK, some additions on custom members. So far we've talked only about standard members. And with that, we did not differentiate iPart and iAssemblies. So what we stated so far applies to iPart as well as iAssembly.

On iPart level, there is the specific variant you can create or define custom members. You need to know Vault does, more or less, make no difference handling custom and standard members, in regards, parent child relationship, as well as file handling-- what we will highlight later on in Chapter 3.

With that, and also with the option we discussed before, consider to split your iPart definitions in several factories. We proceed here and recommend for sure split, custom, and standard member definitions. Inventor is able to combine. And remembering when we introduced our capabilities with Inventor 4, we always showed that we are flexible and can define anything.

But there are some restrictions Inventor puts in place as soon as you combine. You cannot share key as well as custom cells, and generate files from them. So the capability to create our own members upfront, it's completely disabled. And this is reason enough to split. So we have the tools to combine, to aggregate, either on file level or item level.

We knew upfront that we have too much content for this class being covered in one and a half hour. So with that, I just would like to highlight.

Watch out for the handout. There is in detail describe what we additionally share for drawing to differentiate tabular drawings for the entire family and individual drawings for each member. And also, the template with the different title blocks matching both requirements are part of the sample data set or the full Vault environment.

Now we did almost everything about defining. Now let's continue with the first check in. As

Peter continues, you will recognize we apply different categories once a factory and its member gets checked into Vault. And we do this, not only to share different properties per category, the major reason is we apply a specific lifecycle, especially for the factory. Simply based on rules, as Vault is capable to read the subtype, the five classifications. Is it a factory? Is it an iPart or iAssembly factory? Vault reads that.

And based on the category, we recommend once you've created everything-- of course, you've tested the geometry and you've validated that before. But now it's the phase to validate for PDM. And with that, this dedicated lifecycle has a state for validation. And the sample data set we call this for testing purposes. And this does one or two specifics. On one hand, consumers, the designers and the engineers, won't see any factory or member as long as we are working on the file or testing as CAD admins.

The second aspect is the file lock. So we mimic that even as an admin, we don't have edit rights on the factory. And this allows to test everything. Did we miss something in the options we talked before? Inventor will try to save. And if the file is locked, you run into an issue. So this is a good proof before you release the family. To release-- and probably we switch just over to the [INAUDIBLE] example-- the downside on one end, having a dedicated lifecycle is you have two clicks more. You need to select the lifecycle and change state for the factory, plus you'll need to do this for the members as well. But having the advantages I talked before, we think it's worth to have that.

**PETER VAN AVONDT:** Shall we switch? OK, good.

OK, so what I want to do is just do the initial check in of Vault files. Before you check in, you have the ability to generate all members, so they are available into Vault. And the moment you do the check in, you will see that they were all listed into the check in dialog box, OK?

The moment that they are checked in, as Markus stated, you will see that we are categorizing iPart factories. But also if you look at the [INAUDIBLE] tab that we also categorize them as member.

How do we do this? Just shortly showing you how the rules are set up. We have a lot of rules in our test environment, or in our demo environment here. But you will see there is an iPart factory member. And we use a classification, it's configuration factory, as a criteria to find to the factory. The same thing for the member, where we find the classification configuration member. We split also between iAssembly and iPart, depending on the file extension in this Vault.

So just about releasing files here. What we wanted to show you here about releasing the files-- the fact is that you have to do this in a bottom up manner, meaning that you need to start with the iPart and iFactory first, and then just work the whole way up in the whole assembly. So what you can do is select the iPart factories that are on the same level. You can do that in one go and say, OK, I'm just going to change the state.

You will see that I've got the option here to also include the direct parents. In this case, we'll also select the compensator iAssembly. But in that case, it also lists all members in one dialog box. So you can do the release in one go for the flange. I just de-selected the compensator iAssembly.

And then you will see that we have this 2-step. So the first one for the members, where you say, OK, I'm just going to erase all members. And then, in the iFactory lifecycle that's assigned to the iFactory category there, you will see that we have "for testing," as Markus stated, or "in use." And then if we put them "in use," it will be available for everyone. So in that case, the flange assembly and the space assembly is released to manufacturing.

Then the next step is to do just the same for your compensator iAssembly. With the same option on, Include Parents, you will see that starting from your factory member, you get all members also into the dialog box doing the same thing. The [INAUDIBLE] process for the members, and the iFactory lifecycle for putting the compensator iAssembly in use, OK?

Just do this bottom up. Start with the lowest iPart, iAssemblies, and then go up to the levels above this iPart. OK? Good.

**MARKUS KOECHL:** For the initial release, you will agree that this workload is pretty straight forward. But now, we start to discuss about revision cycles. And we are going to share two different approaches, and they are related to your requirements to the scope of the change. So does the change you are intending to do impact all members? So is this change affecting the entire family? This is an universal, or as we call here, the overall revision approach.

If the scope of your change is just out of 100 available members, you need to change or to update two or three, it's probably worse to think about an alternative just to revise these three members. With the next practical example, we share this workflow.

How to start? Your start, of course, with a revision of the factory. But then it's like and try an error process without error, just try. Figure out which members are impacted by your change.

Here, the right slide to my talk. So you watch for changes and Vault helps to watch out which members are impacted by this change. And then just revise these edited member files. In the result, instead of getting an increment in the revision index, overall you will get a differentiated index increase just on the members that were affected by the change.

Let's review how this looks like in real life.

**PETER VAN AVONDT:** OK, so as Markus stated, there are two ways of doing the revision of an iAssembly/iParts. It's the overall, where you do a revision of all members, all parts, however some may not change. And then we call it the smart way of doing revisions or revising this kind of iParts or iAssembly.

So what I want to do is with this compensator example 7 is just make a change on the flange here. And I want to change only two members. So the first step I'm going to do is just say, OK, I'm going to do a change state on the flange example 7.

Once again, bottom up starts with the lowest level and then do the rest in the next step. So as you can see, it also lists-- because I've got this option include parents-- it also lists all the members. In this stage, I'm just going to change the flange and also the compensator in one go, because I know later on, I need to do update the compensator also here. So I'm just going to update it here. Going to open iPart Factory, and doing the changes into the table.

Just notice that I didn't change any member, so all members are now locked or released. So I'm just going to change this value to 10, the number of holes here in the flange. And I'm just going to update it.

The moment I'm going to generate the files, you will see that Inventor will come up with a dialog box. It's not Inventor, it's Vault that's saying that it's failed to check out a member file-- which is normal at that time. On the other hand, it said, I cannot generate a member-- that's not completely true. In the background on the work spaces, it has generated the member files. And it's really good, because this gives it the opportunity to review in Vault what happened.

So if I go to the flange assembly and go to all member files, you see this is the Vault status indicator. It says here I've got some new local files. So now you know which member files are affected by the changes I made in the table. And then I can easily say, OK, I'm just going to change the state of the files. It also indicates where the member files are used. So in one go, I can also change the member files of the compensator iAssembly. So that's something I need for the next step, updating the compensator iAssembly.

And now I can just say, OK, I want to get and do a check out on the files. It will say, oh yeah, do you really want to get the files? No, I generated them already. I just say no. And you will see they are appearing as a green icon here, so they are new.

next step is just to say I'm going to check in the files. You will see these two members are listed also in the check in dialog box. And you're good to go. First step.

Second step, of course, is go to the compensator assembly. We need to go to the iAssembly that's consuming the iParts I've changed. And you see all member files here. First thing I can do here is just to update, synchronize the properties. So it will write back the revision value to the files. And then it's just the same procedure as with the iParts, just get the iParts locally, and then I can start and update the assembly locally.

Next step, generate the files, because everything is updated already-- Inventor does that. And save it. And you can just go into it, check it in, and it will list the Vault members. So bottom up approach. First start with the iParts, then go to the iAssembly. If you have missed iAssemblies and iParts, start with the lowest level and just go through the whole procedure in one go. Once you've done that, then the next step is go back here and just do the release for both of the files. Just change the state.

Well, for testing now, that would be obvious, do the testing. For the demo purpose, I'm just going to put them in use, so we can release everything, and you're fine, OK? As you can see, we're using some B-assemblies and some A-assemblies are now available as member files of the compensator assembly. So it's a smart way of managing the whole iAssemblies and revisioning the whole iAssemblies. OK? Good. Markus, up to you again.

**MARKUS KOECHL:** Yeah. In the summary, they overall just to revise the entire family might be on the first few, the easier approach. It's risk free. You do not need to deal with the option that Peter applied to get the impact. But consider the where used locations. If you revise the entire family, you need probably to revise and update all their used location, regardless that your change did really affect the consuming assemblies or not.

So with that, the smart approach-- smart in terms of it reduces the number of files to be requiring update, because you just changed the affected members. And we think it's worth to train to get familiar with this workflow, because bottom line, you will be much faster and more efficient.

Our last chapter or block, file handling. First of all, most frequent used and always emotionally discussed copy design. And a common statement what we get is as we were saying don't copy factory and member files with copy design, because we will break the links. And their response always is, why? I would like to have an easy approach to create new variants, and therefore I like to copy. But there are good reasons that we don't support that.

Remember, iPart and iAssemblies are predefined libraries. There is a table, and derived from this table we create standard components, standard variants. So with that, of course, you can copy consuming assemblies, but this implicates simply reuse standard content as it is meant for. Reuse library files as well as part family members.

There might be a requirement that you need to create unique members, then of course, you can copy the member file in the assembly context. And if your design intent is really to get a new unique member, then it's fine that we break the link to the factory. And it's starting to get an individual independent file.

Use case number two. You apply copy design. And of course, if you need new families based on the existing one, because they are getting similar geometries et cetera, there is no reason that you could not use. So we recommend you use copy design to make copies of your iPart and iAssembly factories, just as a starting point to create new families.

Some background information or arguments, why what we do and what we don't, and why don't we do everything. So we don't create new variants or members. Because it's a domain of Inventor iPart authoring or the in place design, you can switch in Inventor to create new members on the fly. But the driver of the table is the iPart Author, it's Inventor. So it's up to inventor to create new ones, and to add all new meters, keys, et cetera to your family. You need to edit the table anyway, even if we would have copied it before.

Then once you did create your new, then you can extensively use copy design to consume these new members. Simply apply the replace options, replace existing members, find new ones. So in a graphical view, the workflow is copy your iPart factory, edit the table, create new members, check in, make this available through Vault. And then, again following the bottom up approach, you can also copy iAssemblies that use iParts, replace with the new members, and then open and again, generate the iAssembly member files.

To illustrate this more, when we again switch over to Peter's screen.

**PETER VAN AVONDT:** OK, good. So we have this compensator assembly, example seven again. And as you can see, we've got here to do the two iPart factories. What I want to do is create a copy of them both, and then create a copy of the second level, the compensator assembly and reuse the whole thing. I just want to show the whole workflow to make this happen with a copy design. So the first step I'm going to do is copy the--

**MARKUS KOECHL:** So just summarize, our final goal is having a new independent complete iAssembly family containing new iPart members as well. So to do this in one shot. And this is the final goal.

**PETER VAN AVONDT:** Indeed. Thank you, Markus. So what I'm going to do first is making a copy of this iPart factories. Just copy design, simple the iPart factories. Well, I did some pre-work already, because it was the same workflow. So I'm only going to do the copy of the flange. The space was already in the data set. So you can do that on your own. Or if you really want to, you can use our data there.

I'm going to rename it to example number 8. Good. And what we also did is added an extra rule. You will see that in the hands-on too. If you look at the [? extra ?] rule here, what we're doing is excluding the factory from any property manipulation. All properties should be imposed by the table, like the part numbers. So that's something that you have to do to manage within the table, therefore, we are just removing them from the whole action list.

So just create the copy. Nothing fancy, it has created just the flange number 8 here. Then we are going to open it up into my Inventor environment. And the first thing I have to do is say, OK, in the table, make some modifications. First of all, at the part number, I'm just going to make it example number 8. Update all part numbers in one go. And then you will see that the member name has also been set to example number 7. You can do this one by one.

A better way of doing this is managing it via the spreadsheet. Yeah? You know an iPart or an iAssembly there's always the spreadsheet behind. So you can leverage this functionality just to say, OK, seven minus becomes eight minus. OK. Replace all. And all member names are renamed in one go. Good. So that's the first step.

On the other hand, you can always do some more modifications at members. Make this just more unique or as a new kind of flange may make modifications, of course, in the whole thing.

Just going to make sure to update also the properties, so everything is up to date. And at that time, I can just generate the [INAUDIBLE] members, generate files. Save All. All messages-- oh, this is not good. I need to verify just one thing in the table. Why did it change back? I don't know.

**MARKUS KOECHL:** You're iProperty value got updated.

**PETER VAN AVONDT:** Yeah. OK. Just regenerate it once more. OK. Now everything is fine to check in. And I've got all members created.

First step. We did the same for the space with iPart, and now we can head over to my compensator assembly. in the compensator assembly, I'm just going to do the same thing, copy design. And you will see it will list the spacer iPart factory a few times. And the flange iPart factory also a few times.

First, as a direct link to do your compensator iAssembly. This is the direct link to your table, meaning all values in the space of iFactory are used in the table. So there is a direct link there. Secondly, we have the default members linked also into your iAssembly. And default members are as a file link, as a component link into the assembly, and they're also referencing the whole thing. And so, as Markus stated, we can create a copy of this assembly, and just use the replace to replace all of these factories and members in one go.

So here we go. We're just going to replace the factories. As a tip, you see it will automatically put this is copy. As a tip, we put it also in the documentation-- is in the copy design, you have also to select references. Select references will give you the ability to just make a modification to this one. Now you see that he replaced this one in one go. I don't want this behavior, because I really want to do it one by one. So with the select references, you could avoid this.

So I'm just going to do this for the flange. Replace it, and you will see it will not affect the member file. So now I'm going to replace the member file with the correct one. And it's a spacer, sample 8. This one.

And the last step is the same thing for the flange. I need the member file, it's here. OK, replace. And now I'm almost good to go. Yeah. The only thing I need to do is rename the compensator to 8, as an example. And during the copy design action, it will update all file references in the whole thing.

**MARKUS KOECHL:** Peter, probably at this stage we could share what we discussed creating this example. We really were not happy that we need to select factory members, but there is good reason again. We thought about, can we avoid? But finally, we also recognized we can't avoid. Because as Peter told before, the reference of the factory to the iAssembly is simply to make all variants to publish these to the table as available members of the iAssembly.

And finally, your copy design intent also could be to add additional table variants, and not just replace with the factory. Also, the existing members. Your design intent can be to stay with the existing ones just as additional ones. So our copy design cannot read your mind, your design intent. And with that, we need to do this one by one. There is a question?

**AUDIENCE:** [INAUDIBLE]

**MARKUS KOECHL:** Yeah. To make this question available to everything, the question is which lifecycle applied the copy design to the new created files. Would you like to answer or shall I?

**AUDIENCE:** [INAUDIBLE]

**PETER VAN AVONDT:** It follows the rules, you know? It follows the rules, what I showed before. It followed the rules of the file classification, and that's why it puts it on the lifecycle. And in this case, this is the iAssembly factory. So it just put it as an iAssembly factory. And with this categorization, we also put it in a certain lifecycle, which is the iFactory lifecycle in our case.

**AUDIENCE:** [INAUDIBLE]

**MARKUS KOECHL:** Yeah, it really starts from scratch. We create new files starting within first index a revision, if this is the initial one.

**AUDIENCE:** [INAUDIBLE]

**MARKUS KOECHL:** Yeah. In case you intend to use the copy design to put your target assembly in a new category, you can apply a rule. For instance, if the use cases to make this iAssembly independent to make to a standard assembly later on, you could apply a rule that resets the category.

**PETER VAN** Therefore, you have also different rule sets in the copy a design that can be created and can

**AVONDT:** be put in place. When you do a certain copy action with a certain goal, you just choose another rule set. In this case, we have these iPart/iAssembly rule sets, just to avoid that the iFactory and the iParts factory just changes from category, and so on. So that's good.

Once you created the copy, just open it up. You will get some warnings from Inventor, because of the fact that we are just replacing some references here updating the iAssembly. This is not the iAssembly. Here it is. And then you're fine and good to go.

The first thing you have to do is just have a look at the table. If I go to the table, it will warn me that there are some mismatches, of course, because not all values are updated. We are not able to put any values during copy design into the table. You have to manually do some simplification.

So same thing as with the iParts, just make some changes. In this case, we also set a member name. So if you create a new row, automatically the member name will be indexed and we will update every single thing here. Yeah. We need to make changes to every single member here. You can do this manually. As you can see, we have already this number eight example referenced in. So that's how we define what we need to modify this.

Again, you will get some warnings about having wrong values in. They are not available, these example seven members. So what I'm proposed to do is just, once again, use the power of Excel here just to replace this 7 minus with 8 minus. Well, in this example, of course, it's working fine. In some other examples, in yours, it can be a bit more work. But replacing members is quite faster in Excel than doing the same thing in the table. And then you're done.

If everything is fine, you can just generate the files. OK. And at that time, once generated-- OK. We'll generate all the members in one go. You can check in your file, and you will have every single new independent iAssembly with old members available into your Vault. So that's the way you have to do it. You have to follow this procedure, otherwise you will end up with broken links and so on, OK?

Any other questions about this? No? OK, good. I will switch back over to Markus's PowerPoint. Sorry. We're back.

**MARKUS KOECHL:** We are back. So next, rename. Rename of the factory. And the good news is factory files rename in case the subfolder that Inventor creates is the expected one. So creating members-- the subfolder Inventor creates is always equal to the factory file name. The folder name is the same.

And our Vault is taking care that this rule is also applied in the Vault environment, independent of Inventor. So this means that as soon as you rename the factory, Vault completely reorganizes this entire family. It remains the subfolder as well, and moves and updates all the relationships of the member files within this folder.

So a rename of the factory file means a completely re-organization of your family.

**AUDIENCE:** I'm just wondering what year-- was it this one that changed in [INAUDIBLE]?

**MARKUS KOECHL:** Yes, this is one of the updates or improvements we did with 14 and later. We did not in the past. You're right.

So Vault helps to solve or to prevent you to violate this iPart organization. But in case the subfolder is not following this rule, it's named differently. And there are two major use cases that can happen. For instance, custom members. Inserting a custom member in Inventor, Inventor does not automatically create a subfolder or put it into a folder. The user can create or select a custom folder for that.

So this means Vault on one hand will recognize, well, this is a folder containing member files, but there is no rule applied. It's not the same name as the factory, so I allow to rename. So this works.

The other use case you can end up with different folder [INAUDIBLE] if you got your data set via import. Or quite often, what we see is using peck and go, and later on add the files to Vault. They are no longer structured as Inventor would create the member files. So this can result in different structures, not applying any rule. And as long as you are in this situation, Vault is flexible and allows you to reorganize. But once you reorganize them the way that a subfolder or factory names matches, then we stop. And then Vault starts to handle the family as a whole, as a complete container.

What about your rename members? We do rename standard member files. We allow. The [INAUDIBLE] will run through. And we are even sometimes able to update the table.

So you see here an example. I selected more or less 10 members renamed. And looking later in the Inventor iPart Author, you recognize, wow, all the member file names are written back by the rename command. But this does not work always. I already said, sometimes we are able to do. And this sometimes is just one situation. At least we both know only one. And this is if you have just a new created family.

You checked in 500 members, and then you recognize, oh no, there is a typo in the table, and all the members have this typo now in Vault. In this situation, you could either go back to the iPart Table Author, correct your typo, and recreate the 500 members. Preparing the sample data one family as 500, it took around about 15 minutes runtime. So in this case, it's much faster to use the Rename capability of Vault and let Vault update the table, it's less than a minute.

But as I said, this is the only use case we know that Vault is able to write back. Our overall recommendation is don't rename member files. Because we learned before, we discussed already, that Inventor, iPart, or iAssembly, table, or the author drives the name. So we should not try to take over the lead. So leave it in the iPart table definition.

And we also proposed to lock factories for users, it's our domain as CAD admins. So a user won't be able to rename member files anyway and to write back to the table, so we need to prevent that. And while this is just a recap the situation, there is only one known situation where we are able to write back. So with that, we say even if in theory we are able to rename member files, be careful. And this is just something the CAD admin could leverage creating new families.

What about custom members? To make it short, custom members don't allow to rename in general. And this might be a surprise, because it's a custom member, so you can choose whatever name you want during placement, the initial creation.

So on one hand, we could say, well, let's put this on the wish list. On the other hand, looking how Inventor deals with custom members, he writes back the last used values to the table. And with that, we only could write back the name, but not the parameters. So in theory, this could screw up your entries. On other hand, we see that the last few years since iLogic capabilities really established in Inventor, we see a reduced usage of custom members, because iLogic members share more flexibility for custom modifications.

Last topic, what about move? It's pretty similar as the rename. So as long as your factory name plus the subfolder follows the rule, then we are able and support the move. Simply move the factory and remove the entire family.

If you try individually just to move the folder without moving the factory, we take care and prevent, because we should not split up factory and the member container. They are tied together, so we need to take care of them.

Custom configuration folders need subfolder, of course. As a consequence, what we discussed before, they don't follow a naming convention, a naming rule. And with that, we allow to move within Vault.

There's another minor aspect you probably could have the idea to add additional custom other files to this subfolder. There is no strict reason not to do, so we allow. But we remind the user, you're just about to move any file to a family subfolder, and this could result in unpredicted behavior-- not for you, because you know from now on, with the move of factory, remove the entire family, and your file that you edit will move with the family.

Move of members, it's easy. Members are driven by the family as a whole. So again, we take care that they don't spread, that they don't get different locations in Vault. And with that, we prevent in general to move member files.

Well, I think this is a funny part, because this is try and error. And the reasons are the background behind. You can read in the documentation.

**PETER VAN AVONDT:** Yeah, so I see I've got only two minutes to do some funny stuff here. So good. First thing I want to try is just to rename here a folder. You will see if I rename it, it will fail. And you will see that the configuration member folders cannot be renamed or moved. So it's preventative of doing this. So if you really want to rename a folder, you have to rename the factory file.

Go to the Wizard, just applying that the names [INAUDIBLE]. You will see that the folder will follow directly in here. As said before, renaming member files, you can do it, no worries. But yeah, you have always to pay attention and to review if the family table is updated or not. On initial release, then it's fine. If you do some of the manipulations at that time, this is not working as expected.

It says it's repaired, meaning it repaired the links to all of the family files here. If I open up this part, and I'm just going to verify the table, you will see it will keep the old names here, because it was not in initial release. I did the rename of the factory first, and then the members. Yep, it was not in sync, and then you will get this unexpected behavior.

**MARKUS** Well, but there is one indicator in the Wizard that allows to expect this behavior, but you really

**KOECHL:** need to know about. The Wizard has four steps. And if we are not able to write to the table, we skip the second. So we skip the second. It's a preview of all file relationships, and this is what Peter is going to show with the next example.

**PETER VAN AVONDT:** So here, in this case, it will show the related files. And then I'm quite sure that the files will be updated also within the table of file names. So if I'm going back to the flange assembly here and go to the third one here, and just open it up, you will see in the table hopefully that it has updated all members-- just that it indicated the names there.

Another thing I wanted to show you-- we've got a custom iPart here, a custom member. In that case, you can rename the file or the folder. So that you can rename the folder. But what you will see if you try to rename the member itself, it will fail and it will give you an error message or a warning that it's not going through. We can take the folder, move it somewhere else. So you can just move it to another location.

Something else to share here. If you take an assembly and put it into a member folder, it will always warn you that you are going to move this to a member folder. If you say yes, the assembly will be put in the folder, but with that consequence-- if you move the iPart factory, it will move also the assembly within this member folder. OK, just that you know consequences on the move side. Good.

I will turn over the back to you, because we have to wrap up in three minutes. It's feasible or not?

**MARKUS KOECHL:** Hardly.

**PETER VAN AVONDT:** Hardly, OK. OK, in summary, copy simply reuse members and create new variants, as you did initially just using Inventor, and then replace later on. Rename move. Manage the factory and you are fine. Follow the naming convention and you are fine.

To recap everything, I think Vault really does a good job in an advance task. So managing iAssemblies and iParts is not the easiest task you can imagine in PDM systems. And well, you can leverage and benefit from this following the best practice documentation and tips we shared today.

About factory authoring, there are strict recommendations or highly recommended topics, as well as optional ones to increase metadata and PDM capabilities. The Vault configuration we applied with the download. You probably are able, if you have a professional server and you restore our environment-- you can review in detail what rules we apply to manage the lifecycle with the objective to feel less consumption for the end users, for the designers and engineers.

Probably with the lifecycle under revision, today you got an initiative to rethink your approach, what makes sense for your company, an overall revision or a smart revision approach. And finally, the file handling. It has to be done before a factory is released anyway, but nevertheless, it's probably also an important task for CAD admins to organize or to reorganize. For instance, to move from library folders to standard folders, we are capable to support this type of use cases and design tasks.

OK, Peter already announced that we are going to update the download. So if you already did, please do it again. There are, not only additional tips and tricks, there is an updated link curve for the [INAUDIBLE] number tool and additional stuff.

If you have additional feedback, questions, of course, we are happy to get your feedback. And we are available for the next two hours at the Answer Bar as well if you task or use cases you would discuss offline or in detail, we are happy to discuss with you downstairs at the Answer Bar.

There are other classes this afternoon, early tomorrow morning for Vault. I hope you are aware of that. And one more again, thanks for your great demo.

**PETER VAN AVONDT:** Smoothless. Thank you, Markus.

**MARKUS KOECHL:** Straightforward. Thank you for attending. And we appreciate to see you again in the other classes for the next two days.

[APPLAUSE]