

Assembly Configurations in I-DEAS

Brian Slick

Ferno-Washington, Inc.

bslick@ferno.com -or- brianslick@mac.com

800-733-3766

Premium Partners:



Microsoft

Brian Slick – I-DEAS History

- Co-op with SDRC, Fall 1996 (MS3)
- I-DEAS Instructor, June 1998 – October 2001
 - Design-related topics: Part Design, Assembly, 2D/3D Drafting, Best Practices, Surfacing, Harness, and C3P equivalents
- Contract Drafter → Sr. Project Engineer – Ferno
 - Created hundreds of parts, assemblies, drawings
 - “Assembly Manager” for several large (700+ instance) assemblies
 - User training and support
 - CAD evaluation and testing
- Winner of 2004 and 2005 PLM World “Top Gun” Contests



Ferno-Washington, Inc.

- Privately held, global company
- Multiple product lines in Emergency, Mortuary, Therapy, and Veterinary markets.



FERNO



Overview

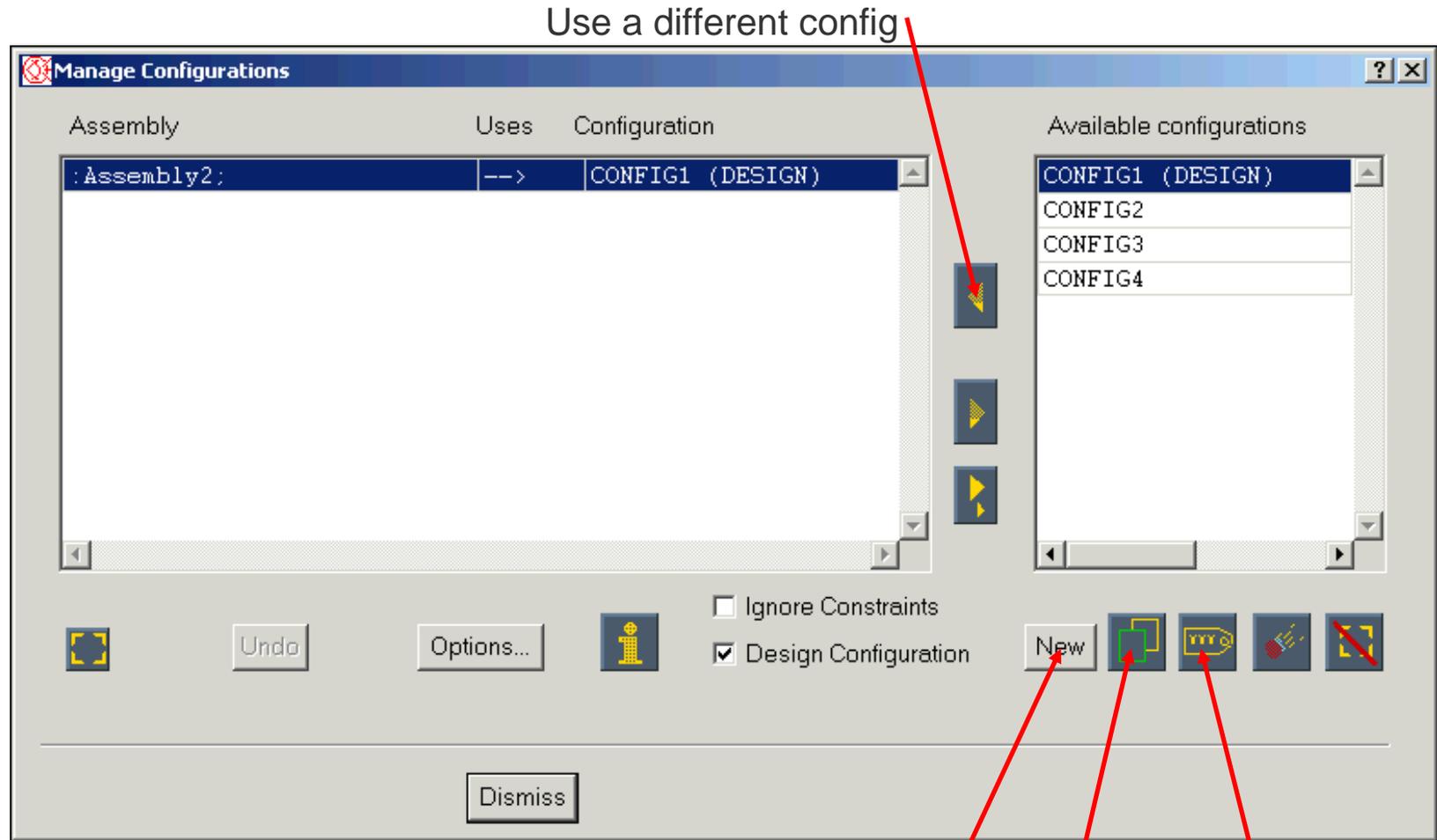
- Basics
- Switching Behavior
- Subassemblies
- Unuse
- Error Messages



Configurations in I-DEAS allow a single assembly to exist with multiple positional and display scenarios. There are a variety of reasons to use configurations, but fundamentally they provide:

1. Different positions
 - Also dimensional values
2. Different displayed information
 - Hide/Show, and Suppress/Unsuppress

Basics – Manage Configurations

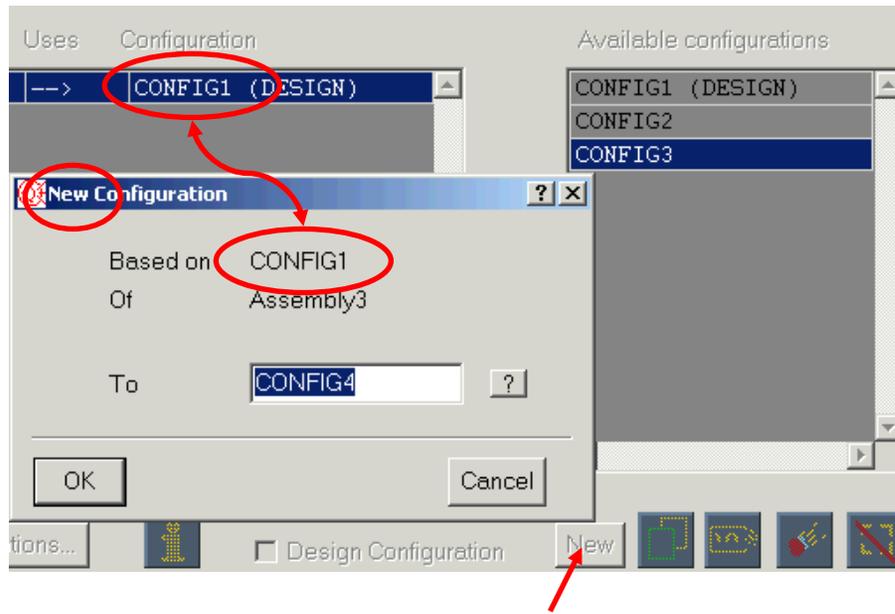


Make a new config

Copy config

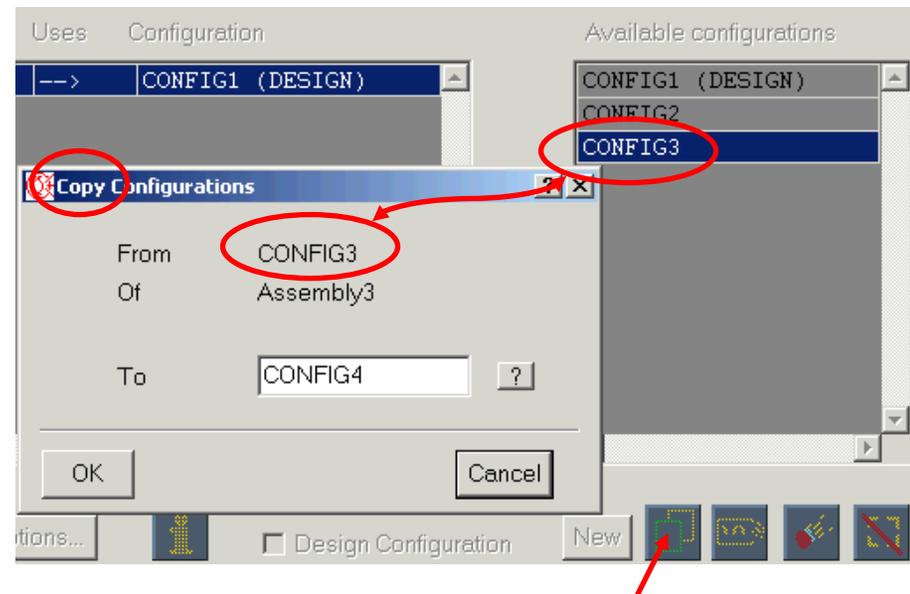
Rename Config

Basics – Creating Configurations



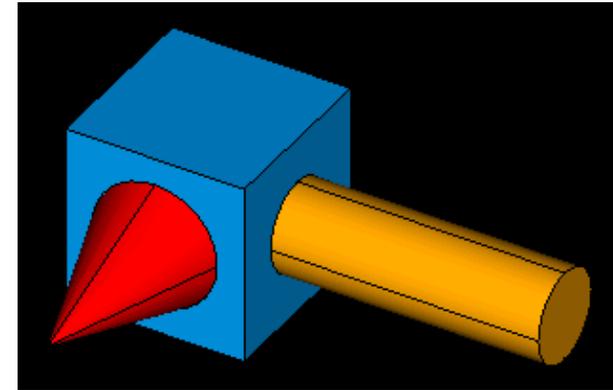
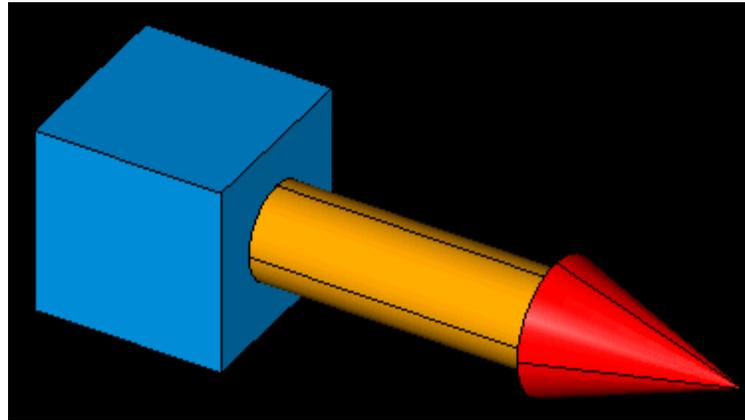
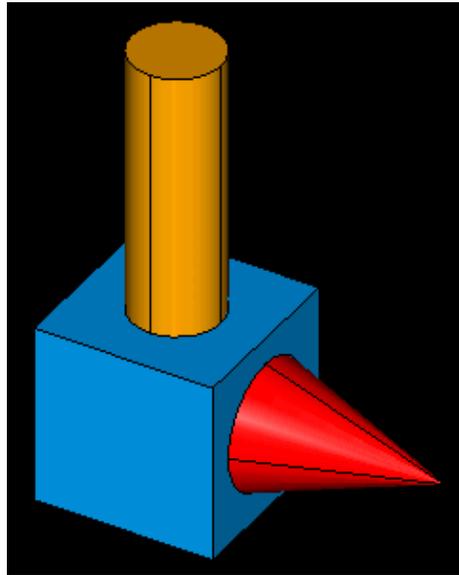
New and **Copy** are very similar. **New** will duplicate the currently active configuration...

...while **Copy** will duplicate the currently selected configuration.

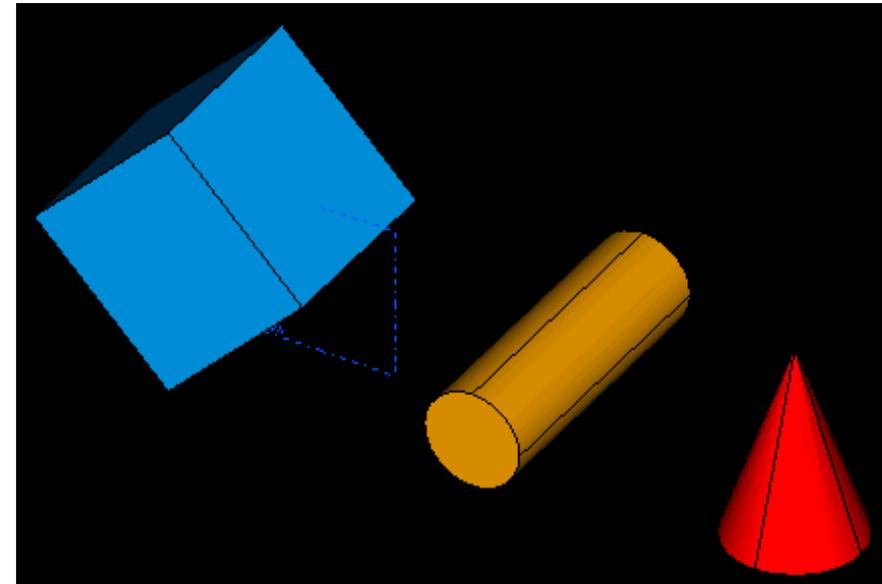


Basics – Positions

Each instance can be in a unique position and orientation in each configuration

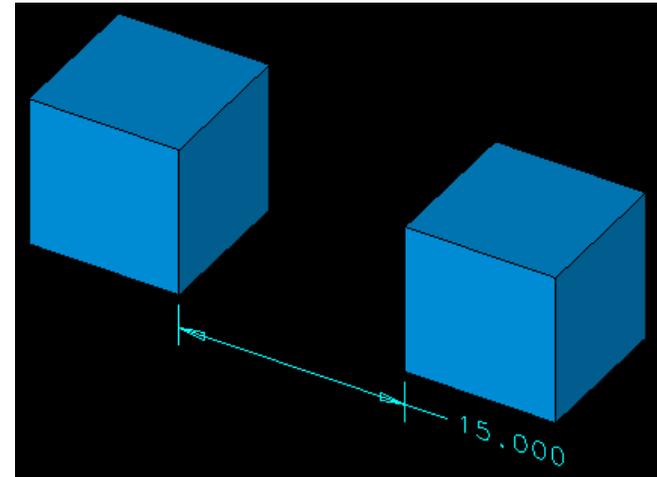
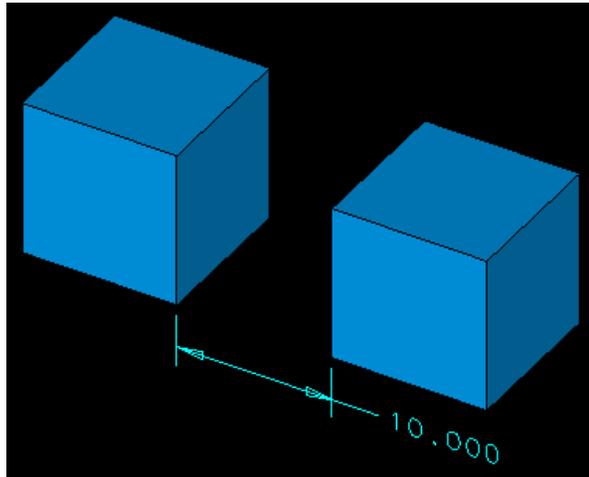
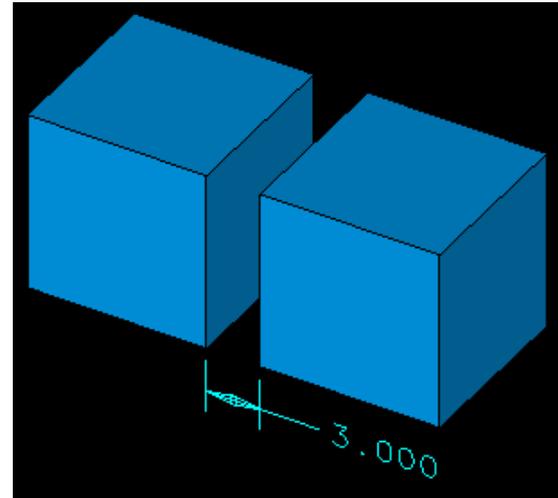
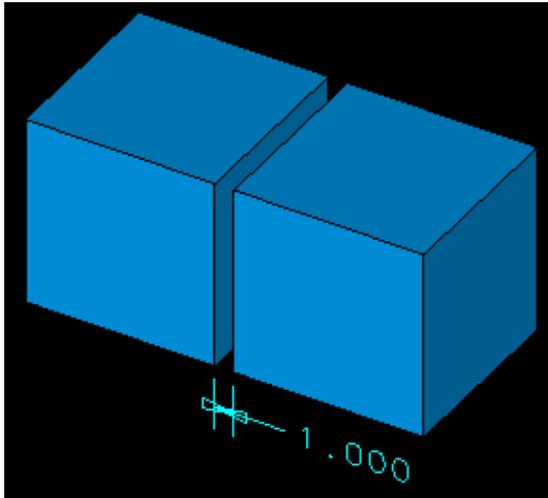


This example does not have any constraints. The part instances can be moved, rotated, aligned, or dragged as the user desires.



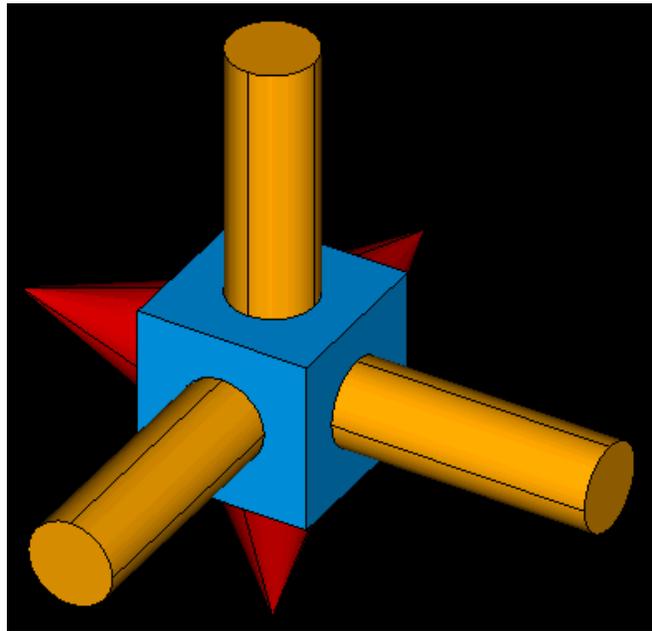
Basics – Dimensional Values

Each dimension can have a unique value in each configuration

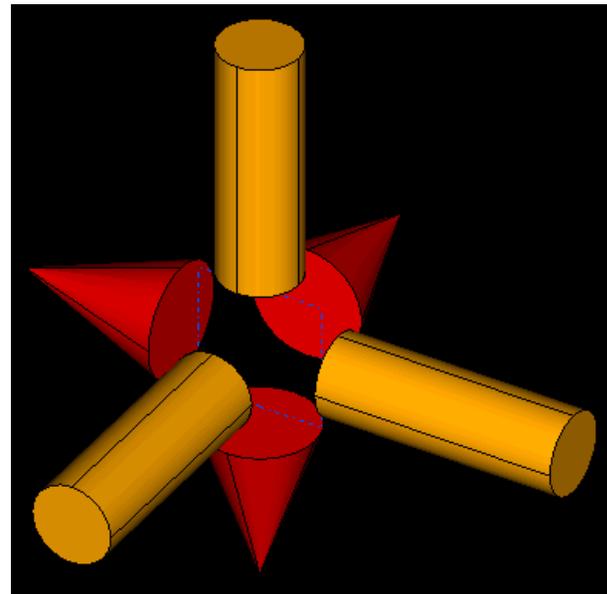


Basics – Hide/Show

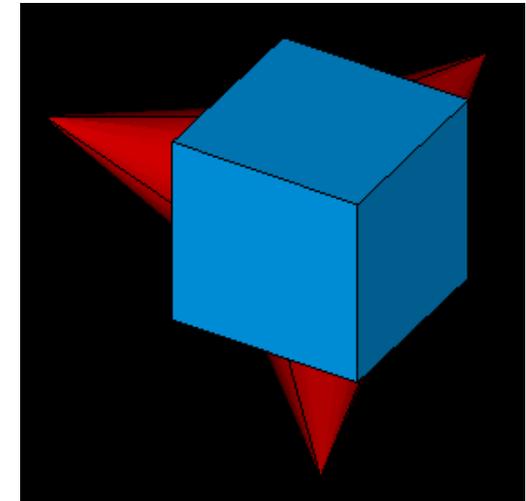
Each instance can be hidden or shown in each configuration



Weight lbf



Weight lbf

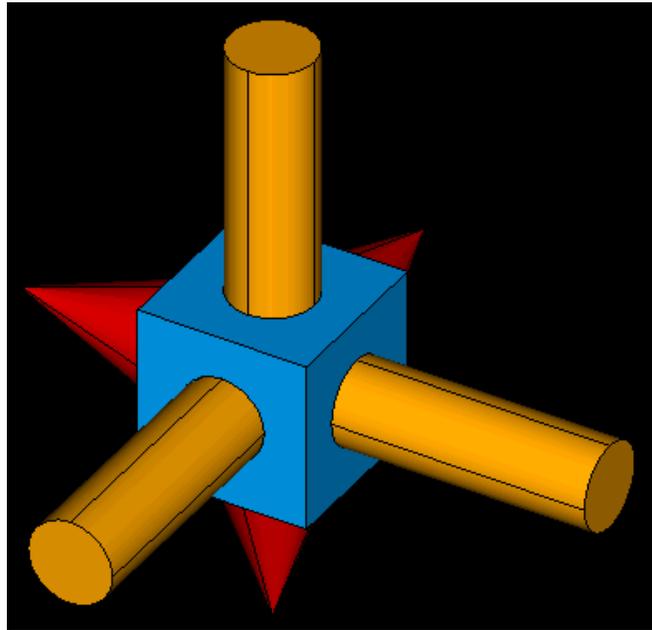


Weight lbf

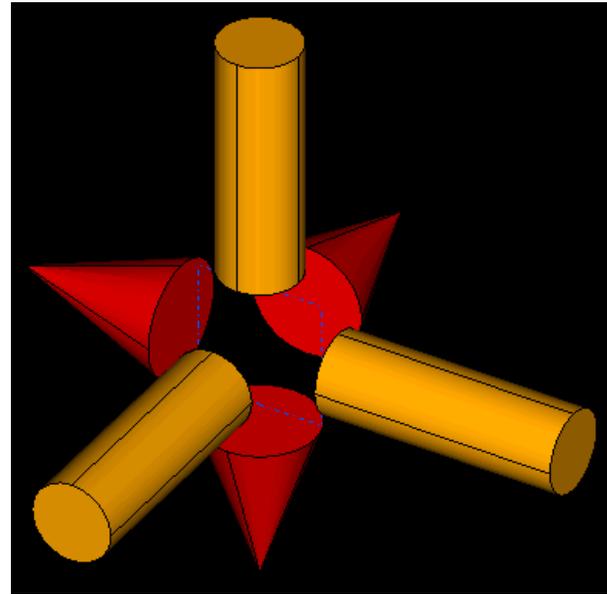
Hide is a display function only. It has no other affect on dimensions, constraints or mass properties

Basics – Suppress/Unsuppress

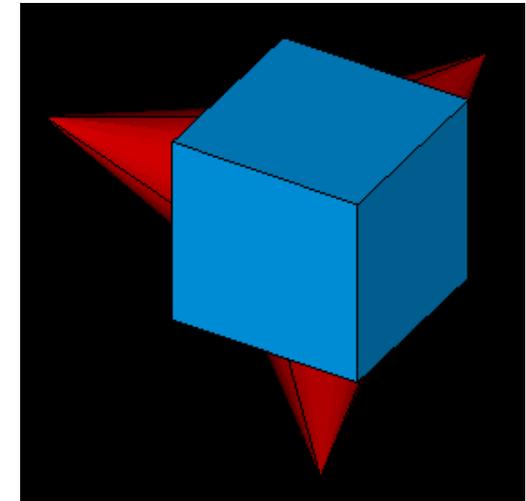
Each instance can be suppressed or unsuppressed in each configuration



Weight 221.322 lbf



Weight 123.755 lbf



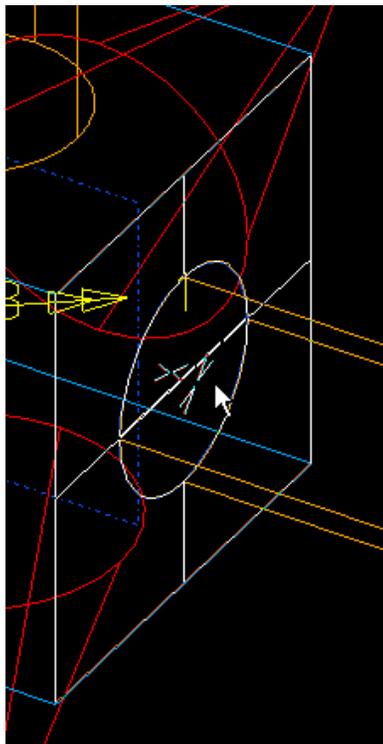
Weight 135.114 lbf

Suppress does have display purposes, but suppressed items do not participate in the constraint network, or contribute to mass properties

Basics – Constraint Behavior

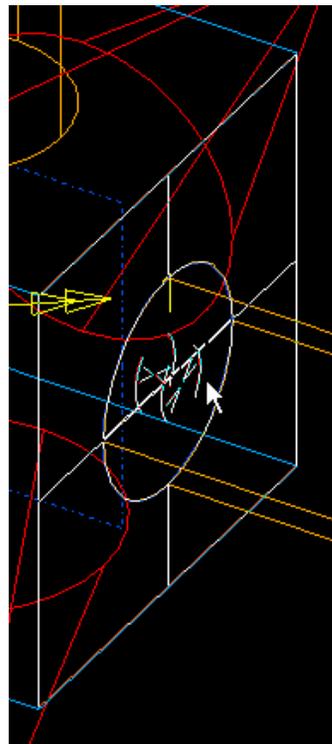
A constraint is either active or disabled across all configurations. It is not possible for it to be active in some and disabled in others. The exception to this is when an instance the constraint is tied to is suppressed – thus leaving nothing to act upon - the constraint is disabled by I-DEAS in that config, but may be active in others.

Active



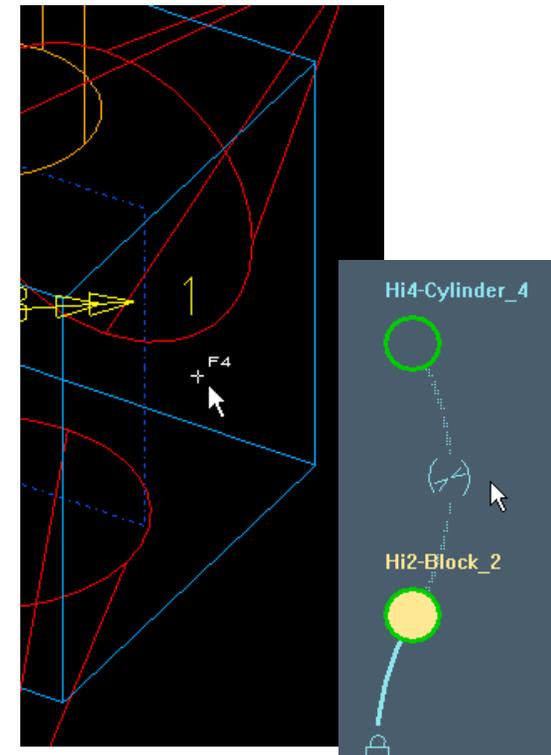
All Configs

Disabled



All Configs

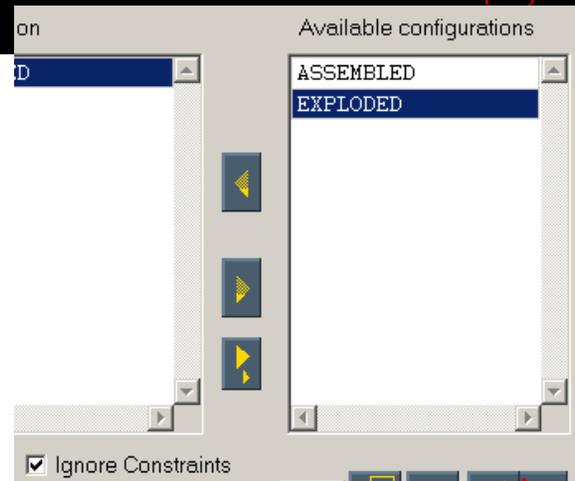
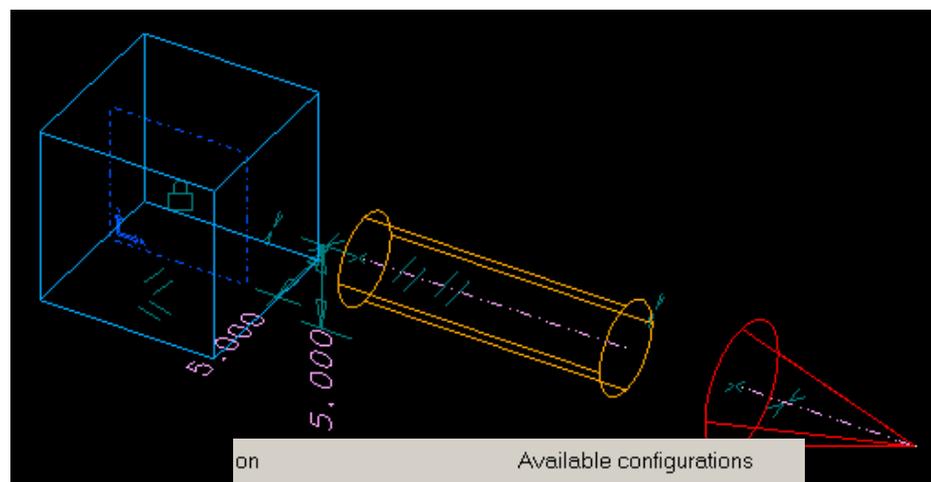
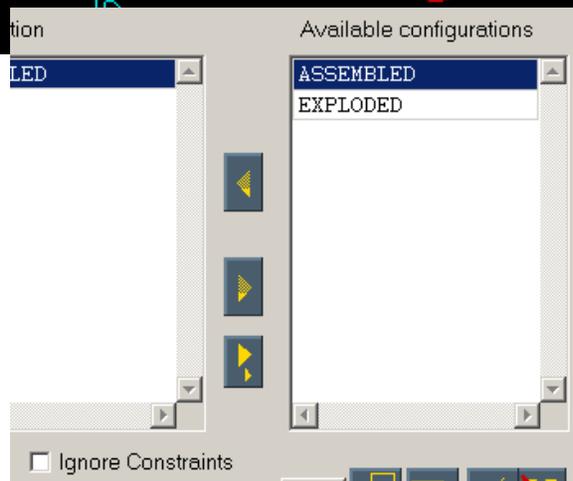
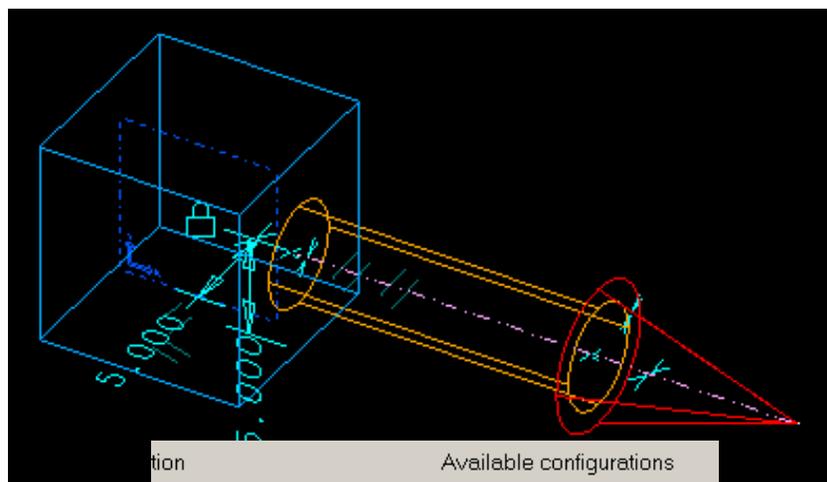
Suppressed Instance



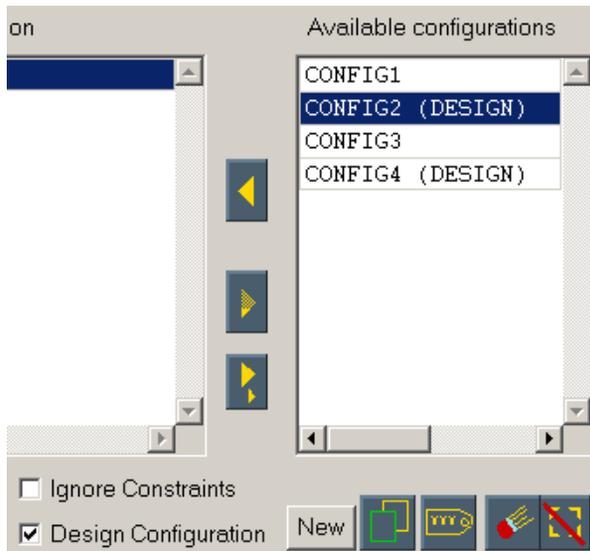
Case-By-Case

Basics – Ignore Constraints

The **Ignore Constraints** toggle does exactly what the name says: does not solve or otherwise enforce any constraints in the configuration. This allows instances to be in positions that their constraints wouldn't otherwise allow, for example exploded for a drawing. Note color differences for the constraints and dimensions.



Basics – Design Configurations



The **Design Configuration** toggle allows additional design intent to be applied to individual configurations. Toggled on, the configuration is marked as “important.”

From a user perspective, this allows anyone who uses this assembly to know which configurations are “good,” and which ones may not be. An example would be any configurations that are directly used to define a print

Any number of Design Configs can be tagged, but the software will offer a harmless warning if the number goes over a default of 2. (Default can be changed)

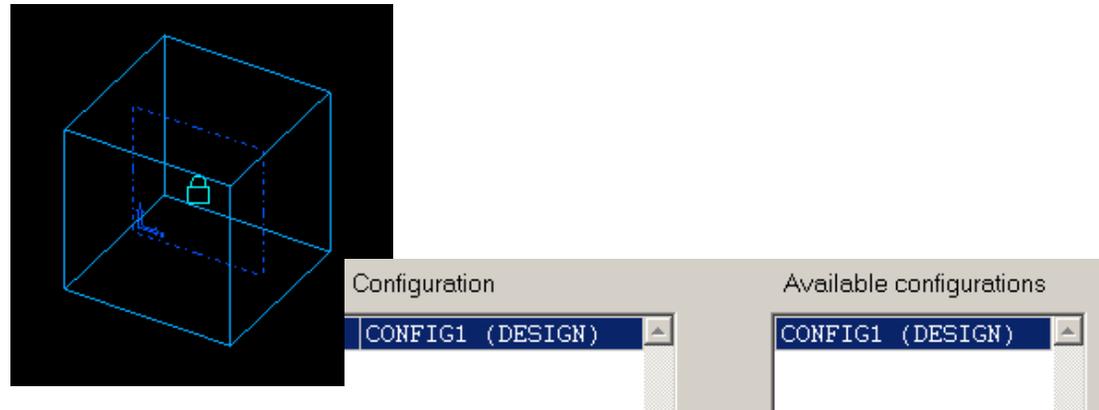
From the software’s perspective, a design configuration has a number of characteristics:

1. If changes have been made to the assembly, ALL Design Configs must be activated (Used) and thus updated before a library Check-In will be allowed
2. When this assembly is instanced into another assembly, the first Design Config listed will be the one that is initially used
3. If a Design Config contains subassemblies, those assemblies must also be using Design Configs or library Check-In will not be allowed

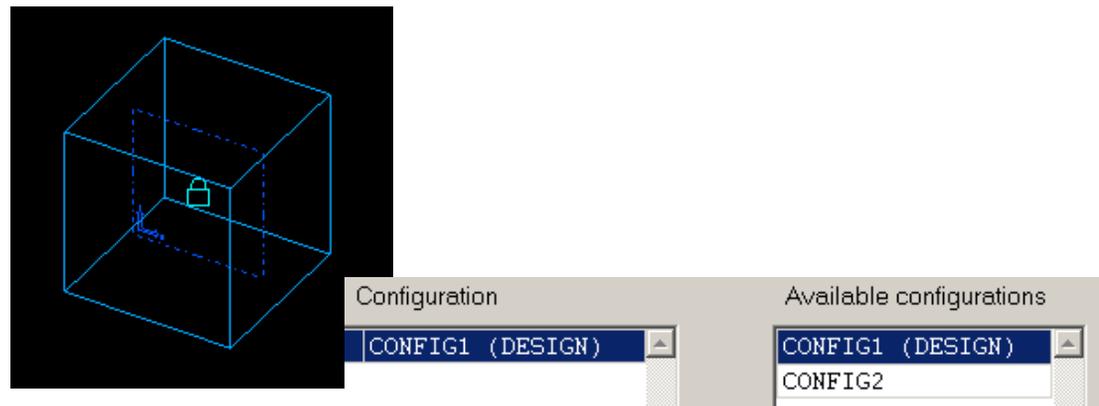
Behavior – Switching Configs + New Instances

If items are added to an assembly, I-DEAS makes some assumptions regarding what should happen to those items when switching to a different configuration. Example:

Step 1: Begin with a simple assembly

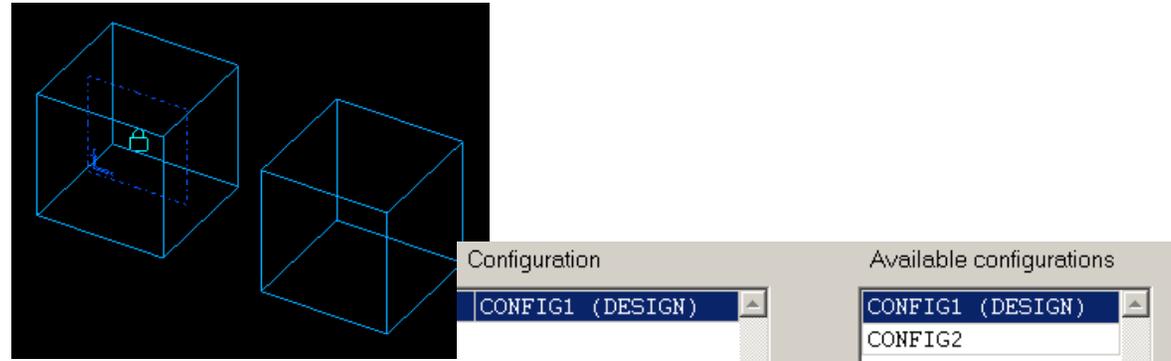


Step 2: Duplicate the config, activate it, then go back to the first one

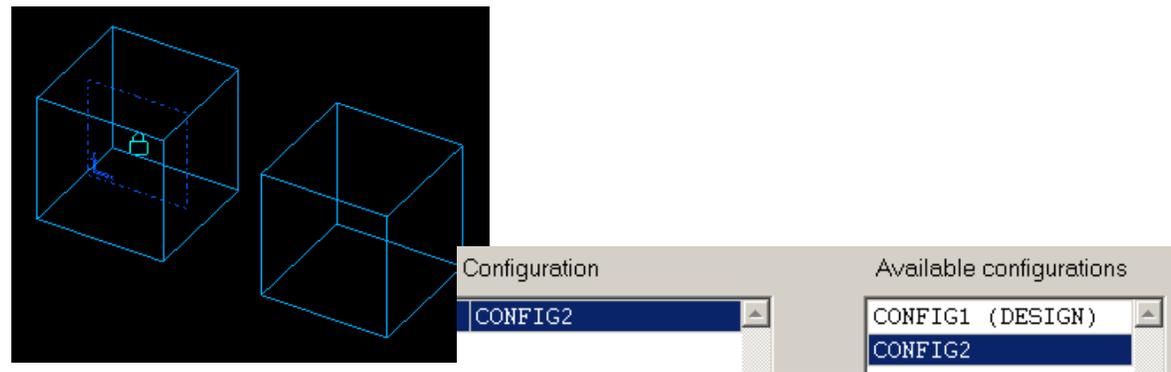


Behavior – Switching Configs + New Instances

Step 3: Add another instance



Step 4: Switch to the other config, and note the List Window

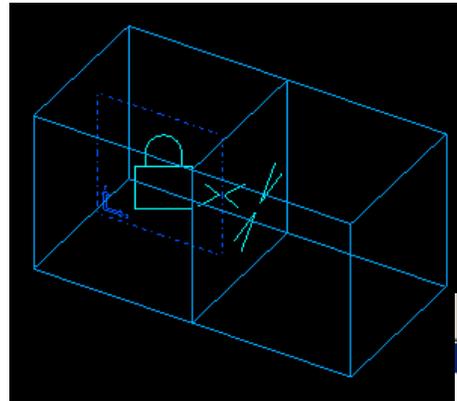


```
--- WARNING ---  
The following instances do not have configuration  
data stored for them in:  
  Hi1-Assembly6_1 (CONFIG2)  
The orientations, dimension values, and attributes  
for these instances are being derived from the  
previously active configuration!  
  Hi3-Block_3 in Hi1-Assembly6_1 (CONFIG2)  
--- WARNING ---
```

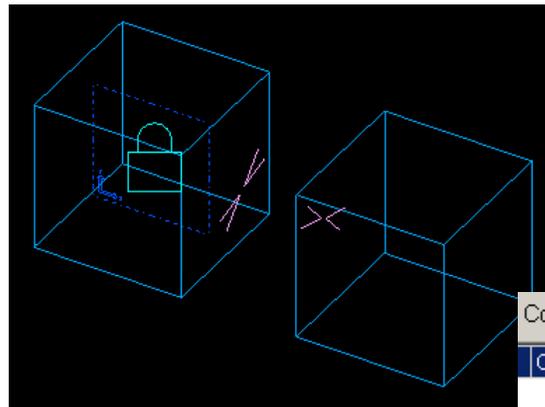
This basically says that because I-DEAS has no information about the new instance in the other config, the instance will be in the same spot it was in the first config.

Behavior – Switching Configs + New Constraints

Step 5: Switch back to the first config and add a constraint



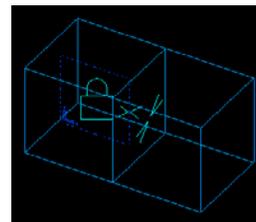
Step 6: Switch to the second config



NOTE: I-DEAS remembers the position of the 2nd instance first and foremost.

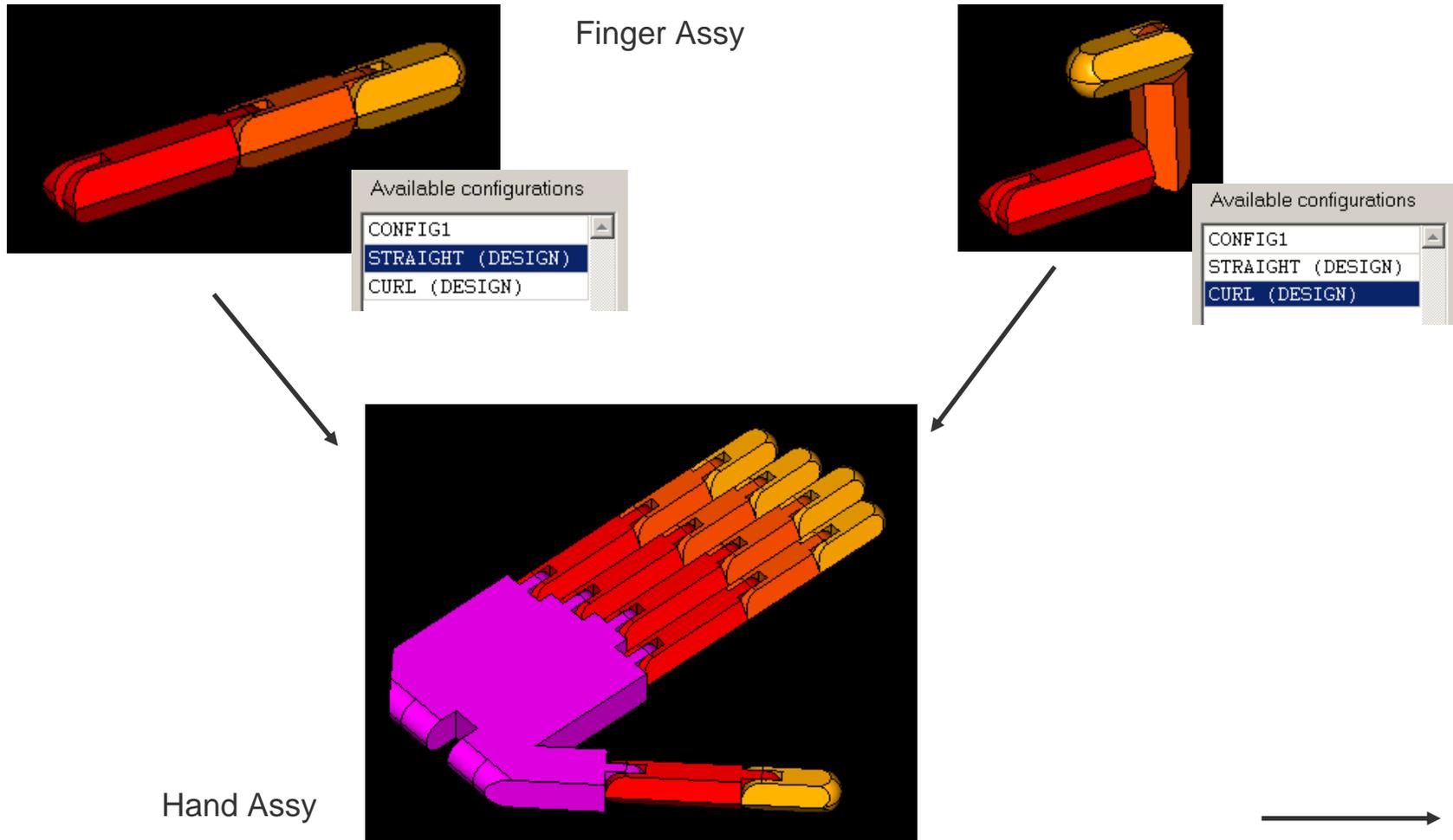


With automatic update turned on, the new instance will slide to its former position. The constraint will be kicked out of date, and the automatic update will then enforce the constraint, pulling the instance back into place



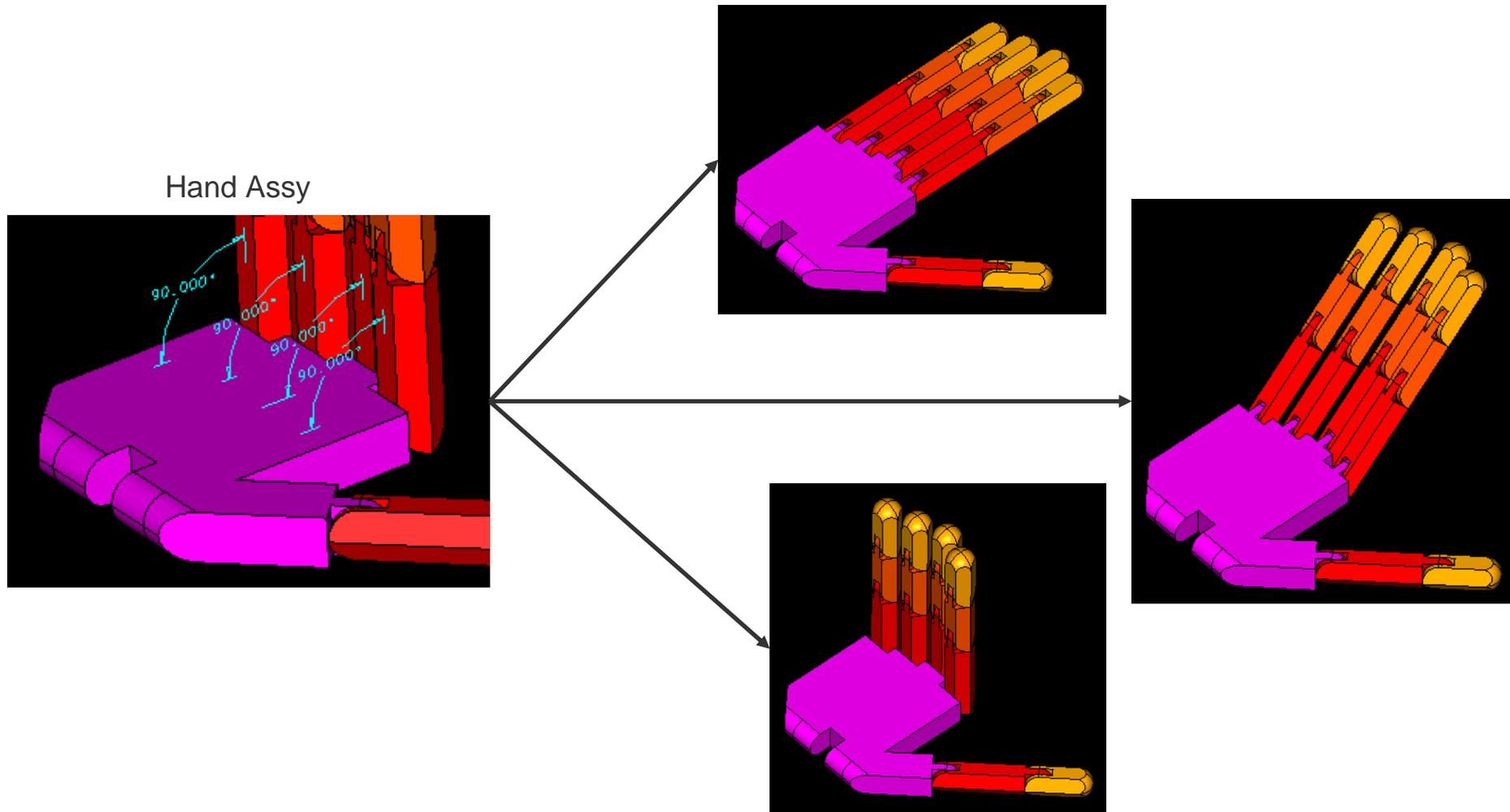
Subassembly Configurations - 1

An assembly's configurations are still available when that assembly is placed into another assembly.



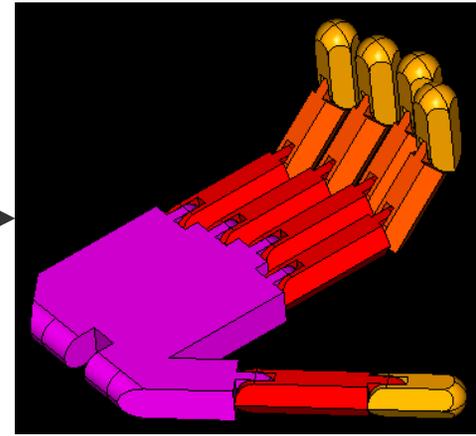
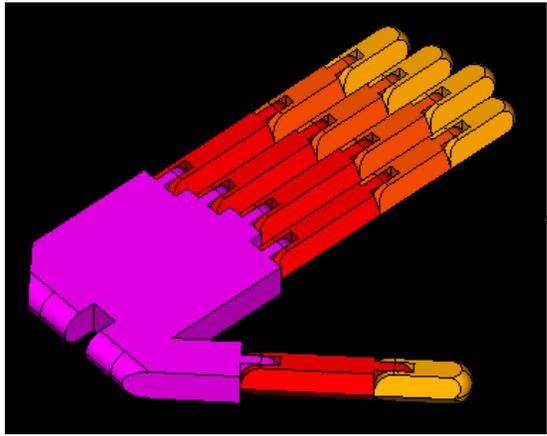
Subassembly Configurations - 2

A parent assembly can use configurations to control dimensional values just like any assembly can, even those that control the orientation and position of subassemblies.

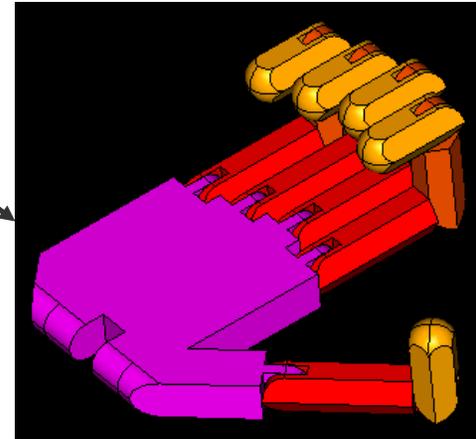
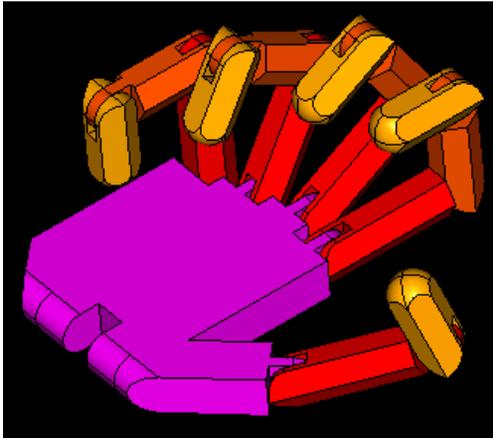


Subassembly Configurations - 3

A parent assembly can also control which configurations the subassemblies will use.



...or control dimensions and configurations



Subassembly Configurations - 4

Available configurations are displayed based on which assembly is selected.

The screenshot shows the 'Manage Configurations' dialog box. The 'Assembly' column lists the following items: ':Hand:', ':Finger;...', ':Finger;...', ':Finger;...', ':Finger;...', and ':Thumb;...'. The 'Uses' column contains arrows pointing right ('-->'). The 'Configuration' column lists 'CONFIG1' for the Hand and 'STRAIGHT (DESIGN)' for all Finger and Thumb subassemblies. The 'Available configurations' list on the right contains only 'CONFIG1'.

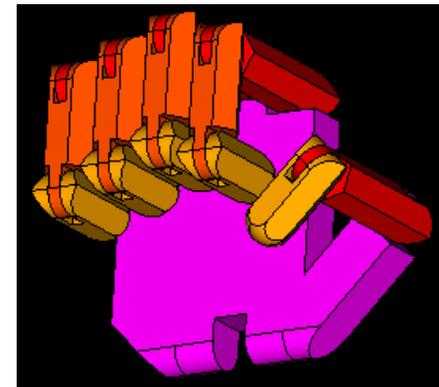
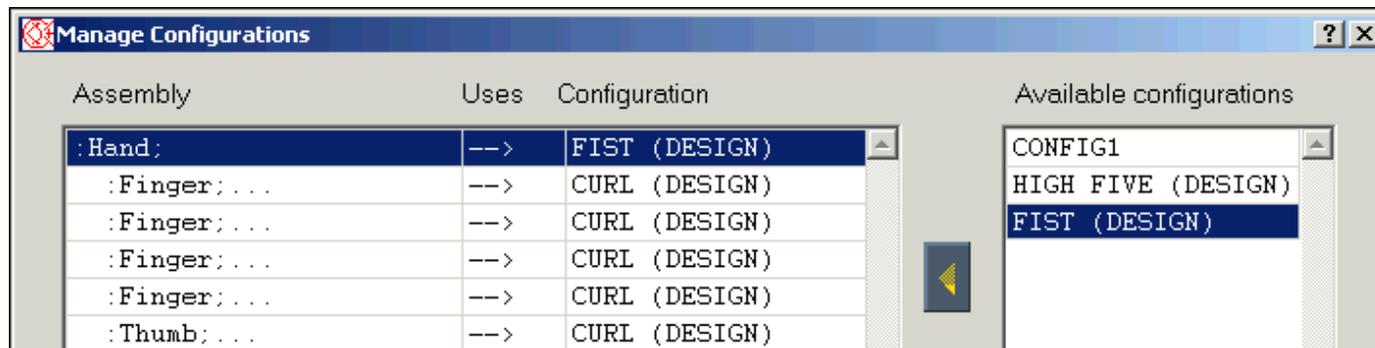
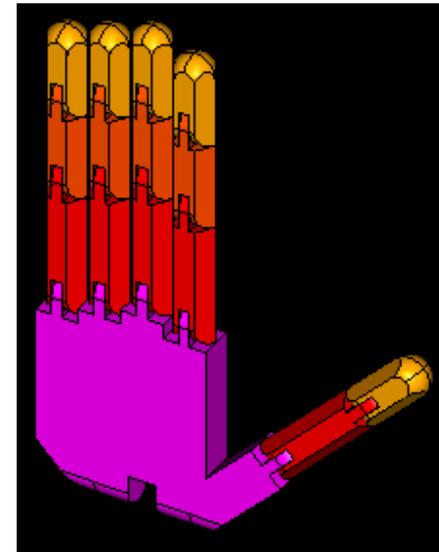
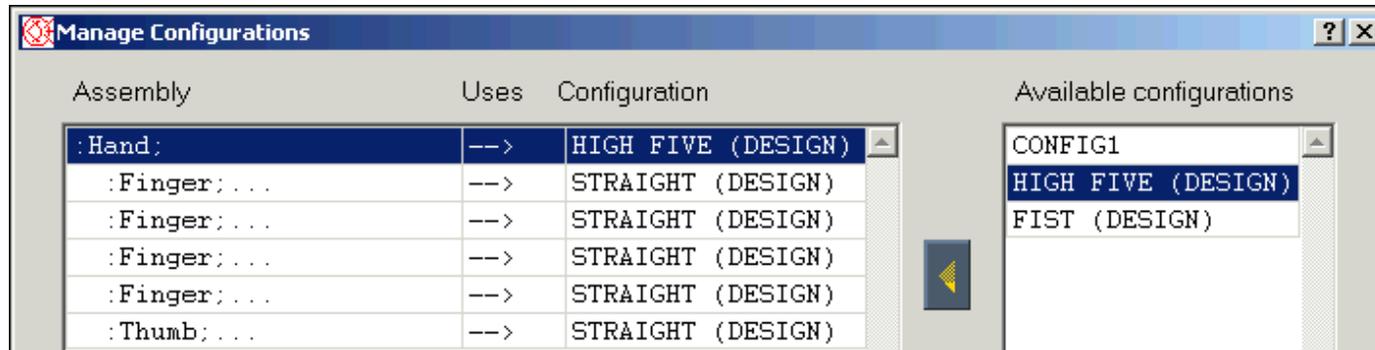
Assembly	Uses	Configuration
:Hand;	-->	CONFIG1
:Finger;...	-->	STRAIGHT (DESIGN)
:Thumb;...	-->	STRAIGHT (DESIGN)

The screenshot shows the 'Manage Configurations' dialog box with the second ':Finger;...' row selected. The 'Available configurations' list on the right now includes 'CONFIG1', 'STRAIGHT (DESIGN)', and 'CURL (DESIGN)'. The 'STRAIGHT (DESIGN)' configuration is highlighted in both the table and the list.

Assembly	Uses	Configuration
:Hand;	-->	CONFIG1
:Finger;...	-->	STRAIGHT (DESIGN)
:Finger;...	-->	STRAIGHT (DESIGN)
:Finger;...	-->	STRAIGHT (DESIGN)
:Finger;...	-->	STRAIGHT (DESIGN)
:Thumb;...	-->	STRAIGHT (DESIGN)

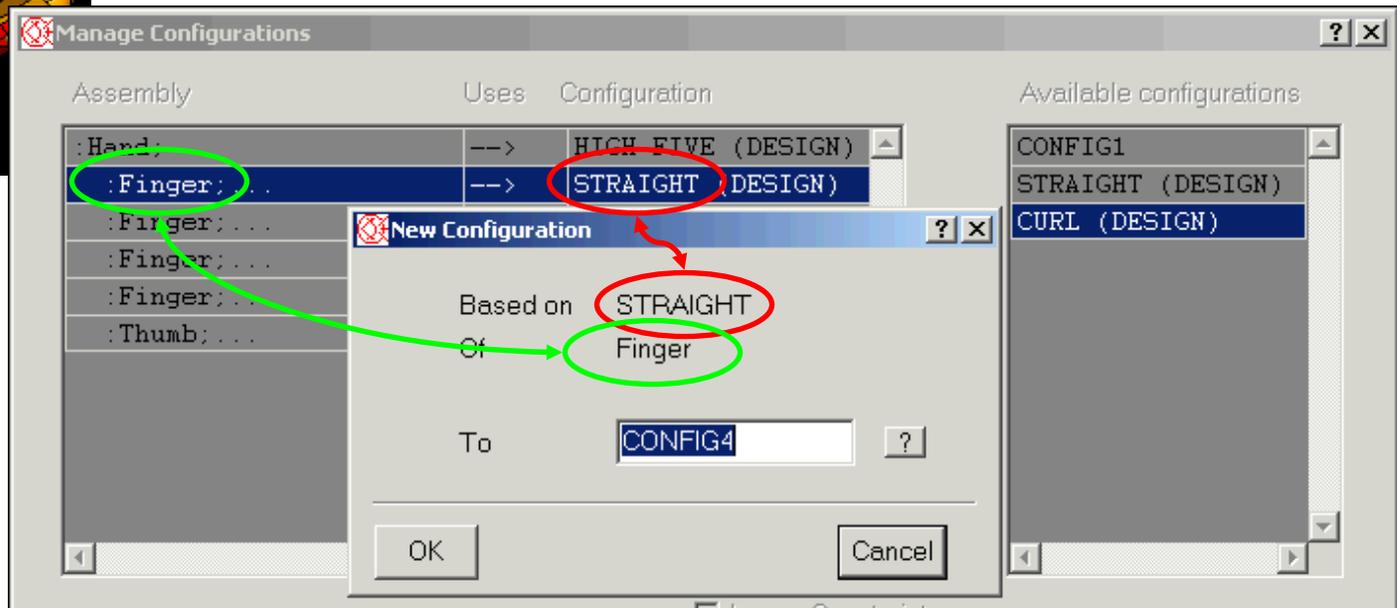
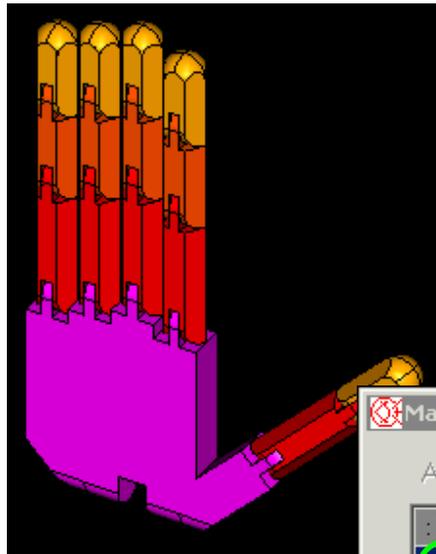
Subassembly Configurations - 5

A parent assembly can have multiple configs that use different subassembly configs.



Subassembly Configurations - 6

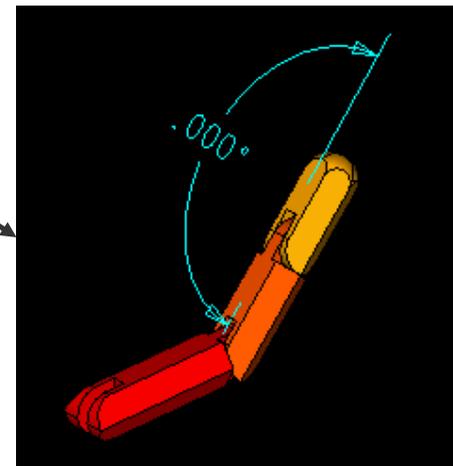
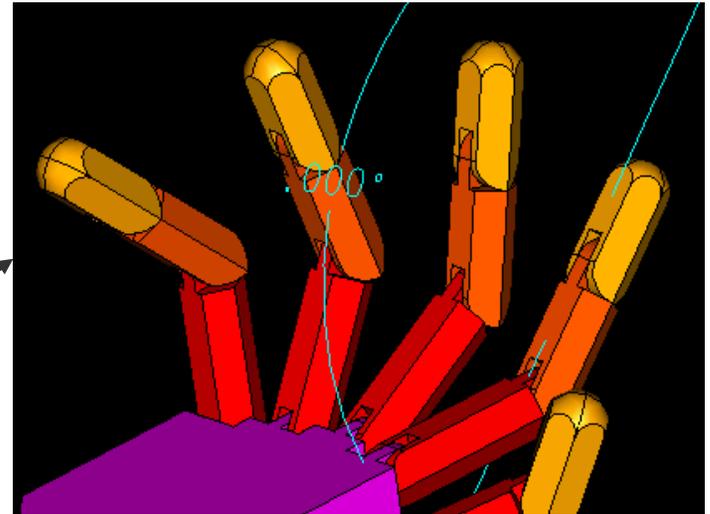
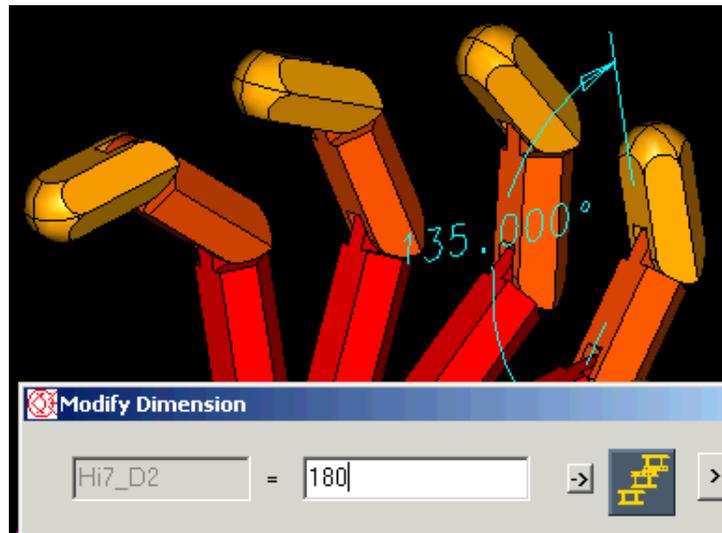
Subassembly configurations can be created while the parent assembly is on the screen ★



★ - if library permissions permit

Subassembly Configurations - 7

Subassembly configurations can be **modified** while the parent assembly is on the screen ★

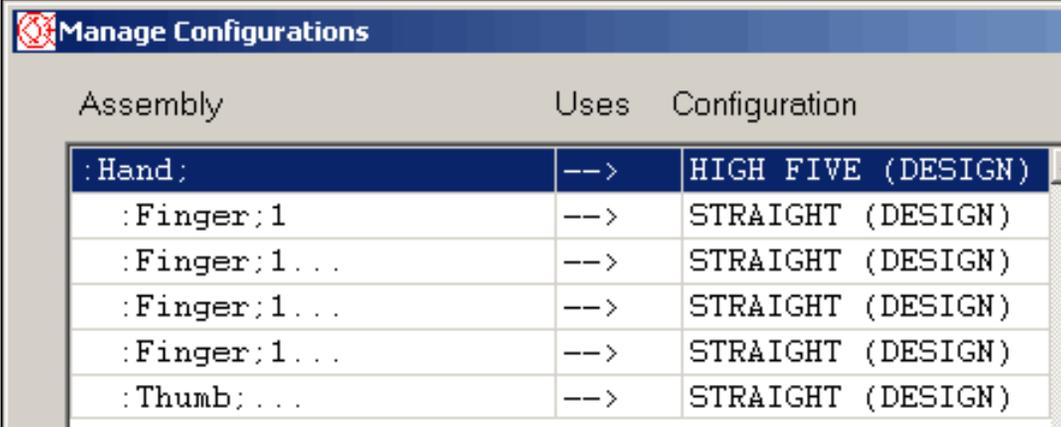


★ - if library permissions permit

Subassemblies & Design Configurations – 1

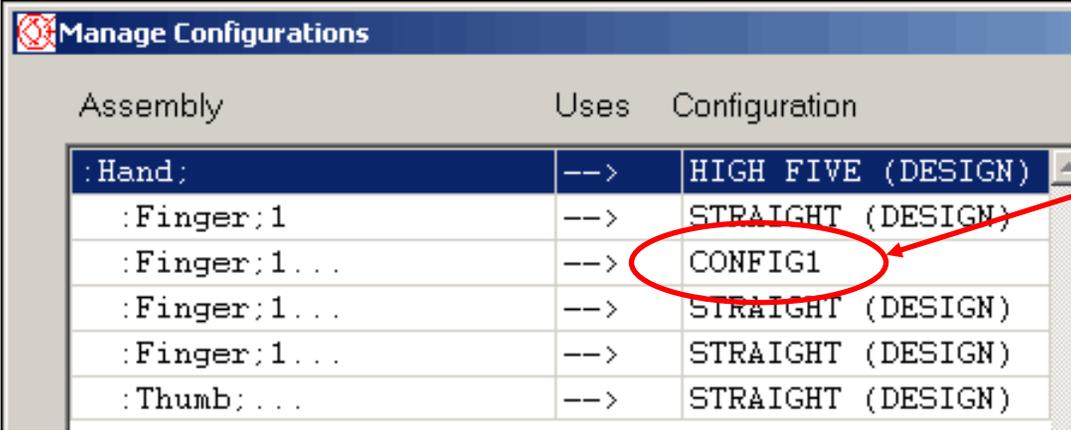
If a parent assembly configuration is tagged as a Design Configuration, then all subassemblies must also be using Design Configurations, or the parent cannot be checked into a library.

Good:



Assembly	Uses	Configuration
:Hand;	-->	HIGH FIVE (DESIGN)
:Finger;1	-->	STRAIGHT (DESIGN)
:Finger;1...	-->	STRAIGHT (DESIGN)
:Finger;1...	-->	STRAIGHT (DESIGN)
:Finger;1...	-->	STRAIGHT (DESIGN)
:Thumb;...	-->	STRAIGHT (DESIGN)

Bad:



Assembly	Uses	Configuration
:Hand;	-->	HIGH FIVE (DESIGN)
:Finger;1	-->	STRAIGHT (DESIGN)
:Finger;1...	-->	CONFIG1
:Finger;1...	-->	STRAIGHT (DESIGN)
:Finger;1...	-->	STRAIGHT (DESIGN)
:Thumb;...	-->	STRAIGHT (DESIGN)

Not using a Design Config

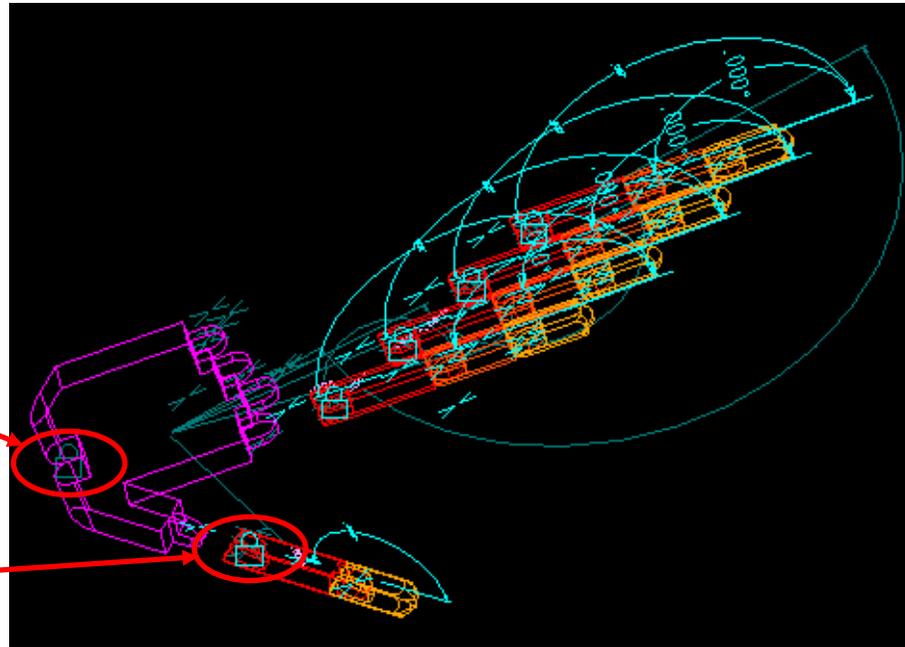
Subassembly Configs & Library Permissions

Switching between subassembly configurations does not change the subassembly. The data is already there, the user is simply looking at a different subset of that data. Therefore, a parent assembly can toggle between a subassembly's configurations regardless of Library Status (Rfl, CK, or Co) of the subassembly. Switching subassembly configurations *is* a change to the parent assembly.

When subassembly configurations are used, the subassembly data is accessed directly. The parent assembly really doesn't know anything about the subassembly, other than the fact that it is present in the assembly, and which configuration is being used. Any changes that are made to the subassembly while in context of the parent – modifying dimension values, adding or removing constraints, etc. – are made directly to the subassembly, even though it is not currently on screen. Therefore, proper Library Status (CK or Co) for the subassembly will be required in order to make the change.

Subassemblies & Ignore Constraints

Toggling on Ignore Constraints in a parent assembly configuration affects only those constraints controlled by the parent assembly. It does not affect subassembly constraints.

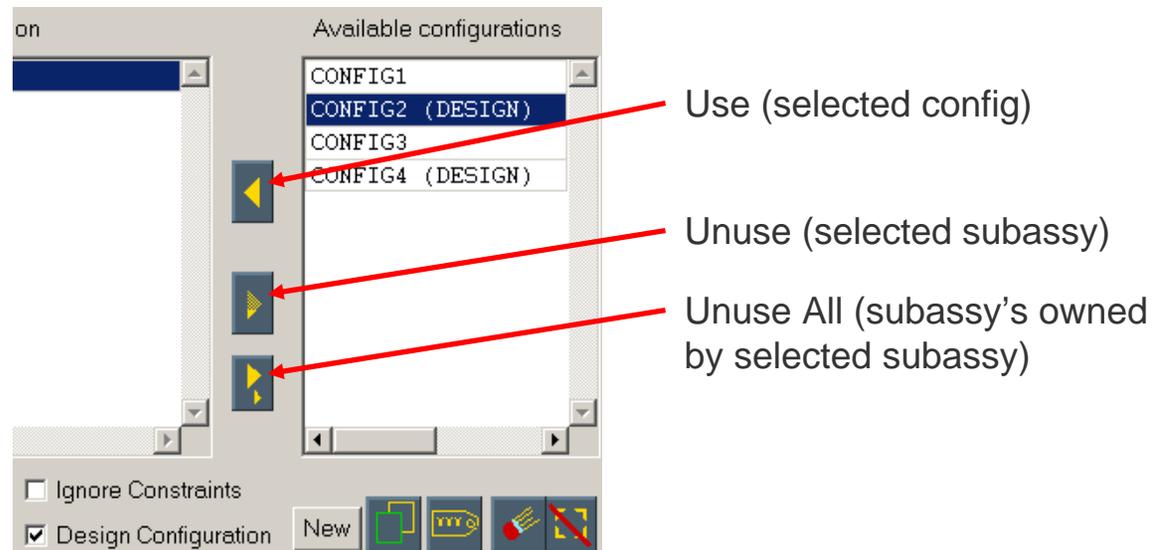


Note the color differences of the constraints. Those owned by the parent are disabled. Those owned by the subassemblies are still active. Pulling the subassemblies apart will simply cause the pieces to snap back into place after an Update.

Unuse - 1

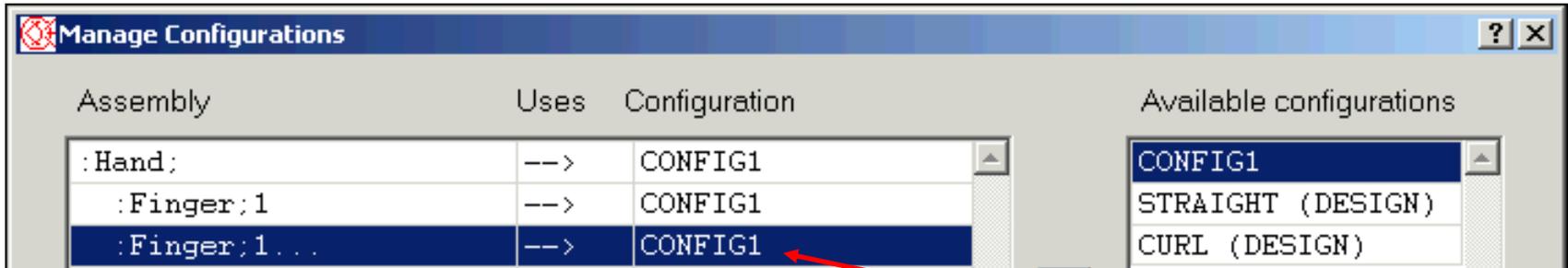
In order to understand what it means to **Unuse** a subassembly, it is useful to review how assemblies in I-DEAS work.

When adding components to an assembly – parts or other assemblies – the real item itself is NOT used. I-DEAS creates what is called an **Instance**. An instance basically points to the real item. An instance is a single item, regardless of what it is pointing at. A single part instance points at one part; a single assembly instance points at one assembly. Even if the assembly being pointed to contains hundreds of other instances, the current parent assembly sees only a single entity: an assembly instance.



Unuse - 2

Unusing a subassembly has a very specific meaning: pretend the subassembly isn't there, and act as if the components of that subassembly were placed into the parent assembly directly. As far as the instances and constraints are concerned, Unusing is roughly equivalent to using **Change Parent** to move the instances higher in the hierarchy. The parent assembly now owns, and is thus able to modify, any dimension values independent of the subassembly. Any newly created constraints and dimensions will be owned by the parent, and therefore never seen by the sub.



Assembly	Uses	Configuration	Available configurations
:Hand;	-->	CONFIG1	CONFIG1
:Finger;1	-->	CONFIG1	STRAIGHT (DESIGN)
:Finger;1...	-->	CONFIG1	CURL (DESIGN)

Config is used

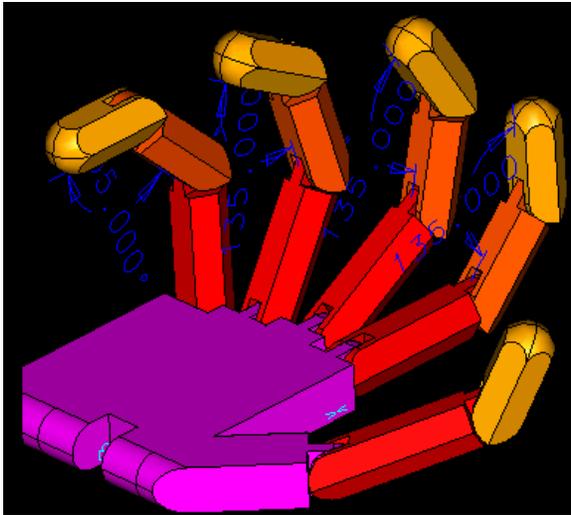


Assembly	Uses	Configuration	Available configurations
:Hand;	-->	CONFIG1	CONFIG1
:Finger;1	-->	CONFIG1	STRAIGHT (DESIGN)
:Finger;1...		from Hand (CONFIG1)	CURL (DESIGN)

Config is not used, controlled by parent

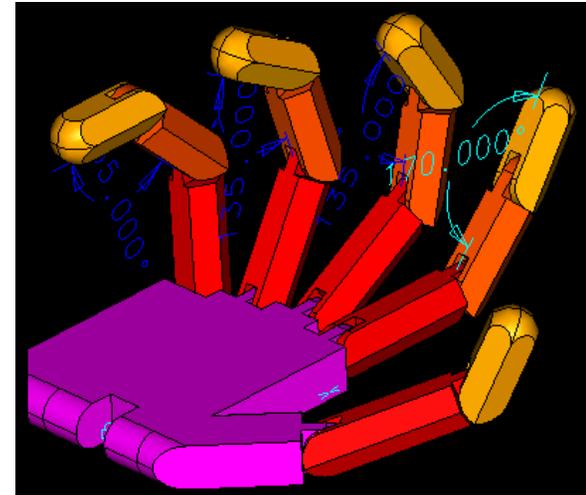
Unuse - 3

Unusing a subassembly can be handy or even necessary if the subassembly does not contain a configuration with the desired position and/or dimensional value.



Subs are Rfl, and configs are used.
Dims cannot be modified.

Unuse

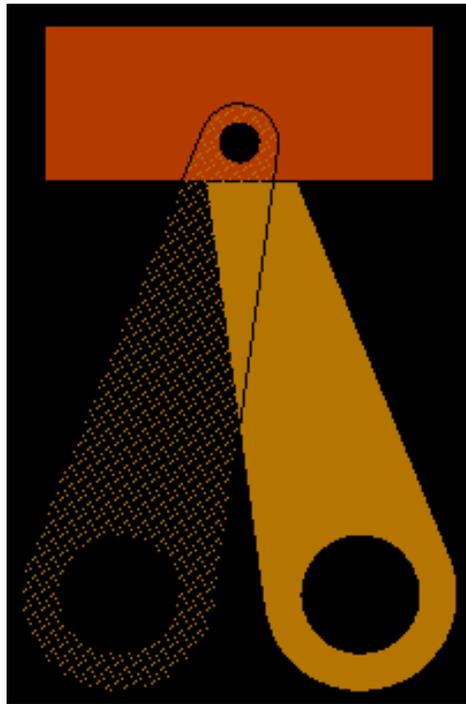


Sub config is unused, parent is now able to modify dimension value. Sub is not aware of the change

Keep in mind that the subassembly knows nothing about the change (it is still Rfl, and therefore not modifiable). If the change is that important, it really should be applied to the sub directly. This sample scenario is “emergency only,” not a “first response.”

Unuse - 4

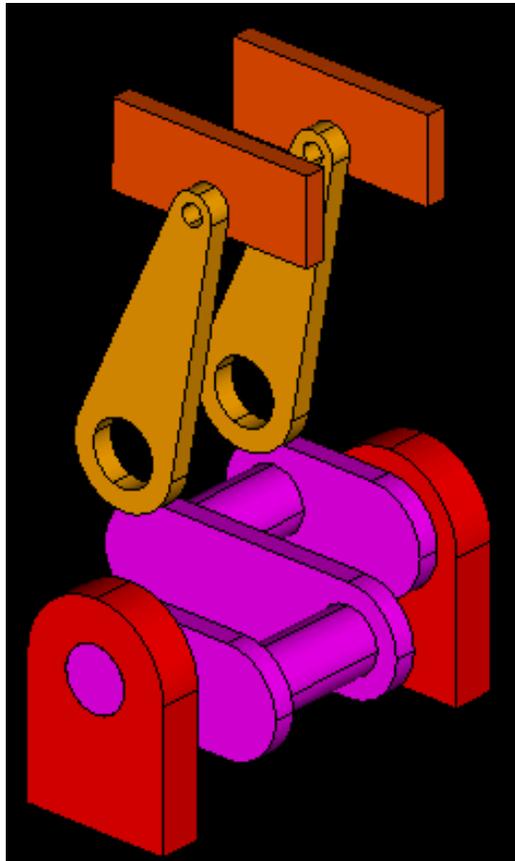
If motion is desired, there may be no choice other than to unuse a subassembly. Consider a simple piston assembly:



Connecting rod needs to be free to pivot relative to the head.



Unuse - 5

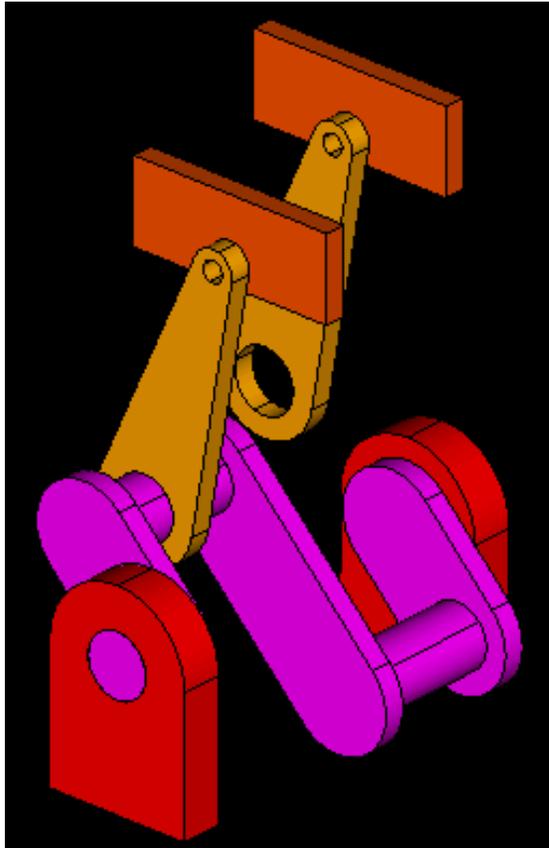


With duplicate instances, only one piston position is possible if both are using the same configuration, or at most two positions if they use different configurations.

Manage Configurations		
Assembly	Uses	Configuration
:Engine;	-->	CONFIG1 (DESIGN)
:Piston;...	-->	CONFIG1 (DESIGN)
:Piston;...	-->	CONFIG1 (DESIGN)

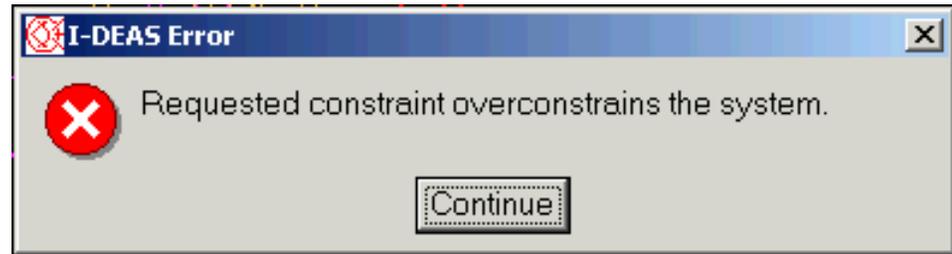


Unuse - 6



The first piston constrained will force the crankshaft to adjust to honor the constraint, if it is otherwise free to do so.

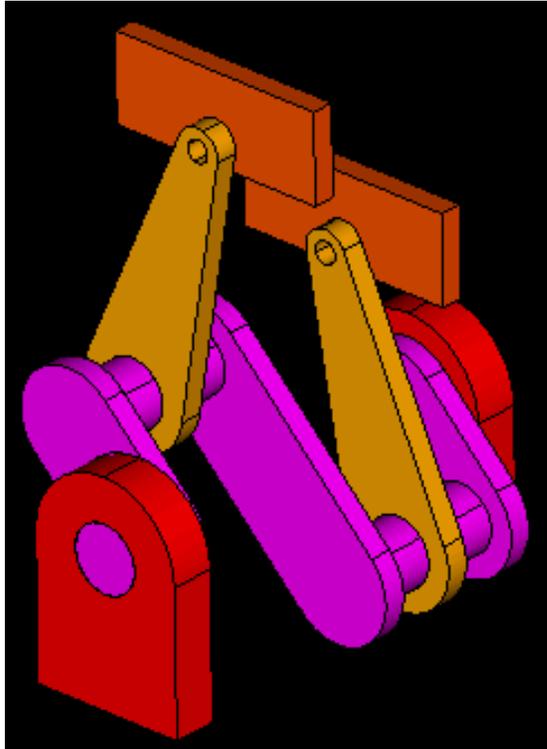
However, the second piston cannot be constrained:



It is not physically possible for both pistons to be using a single configuration, and exist with two different positions. Also, the crankshaft is not able to rotate. The first piston is forcing it to stay at that particular angle.



Unuse - 7



By unusing both subassemblies, I-DEAS no longer treats each piston as a single entity. Instead of solving for 2 pistons, it now solves for 4 items: 2 heads and 2 rods.

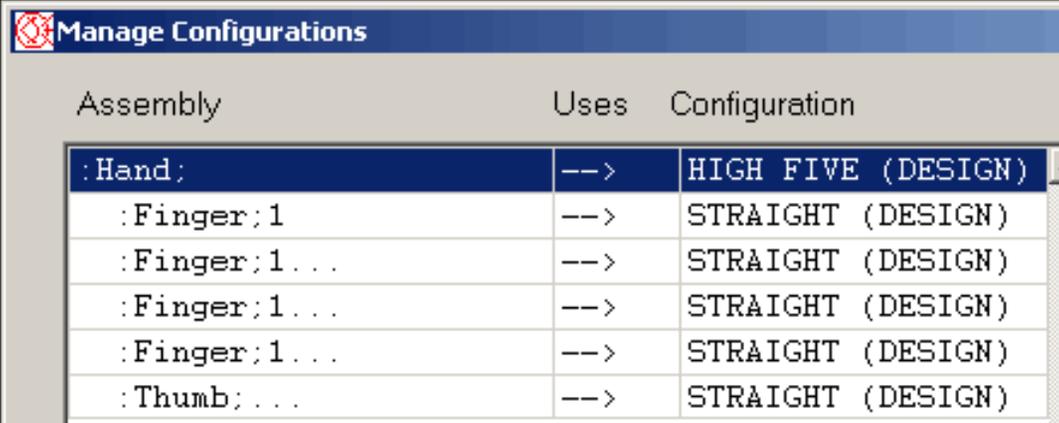
Assembly	Uses	Configuration
:Engine:	-->	CONFIG1 (DESIGN)
:Piston;...		from Engine (CONFIG1)
:Piston;...		from Engine (CONFIG1)

If the piston subassembly was constrained in such a way that motion was not possible (rod parallel to center of head, for example), then unusing configurations will not “solve” the “problem” of being unable to constrain both pistons to the crankshaft. If motion will be necessary at a higher assembly level, then the motion has to be possible at the lower level as well. Basically, the sub cannot be fully constrained.

Subassemblies & Design Configurations – 2

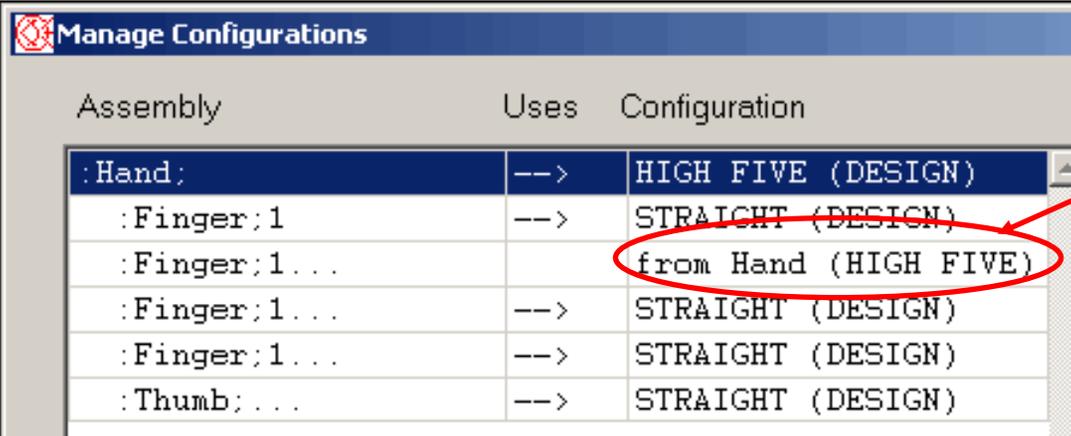
As previously stated, if a parent assembly configuration is tagged as a Design Configuration, then all subassemblies must also be using Design Configurations, or the parent cannot be checked into a library. However, if a subassembly has been unused, then check-in can proceed.

Good:



Assembly	Uses	Configuration
:Hand;	-->	HIGH FIVE (DESIGN)
:Finger;1	-->	STRAIGHT (DESIGN)
:Finger;1...	-->	STRAIGHT (DESIGN)
:Finger;1...	-->	STRAIGHT (DESIGN)
:Finger;1...	-->	STRAIGHT (DESIGN)
:Thumb;...	-->	STRAIGHT (DESIGN)

Acceptable:

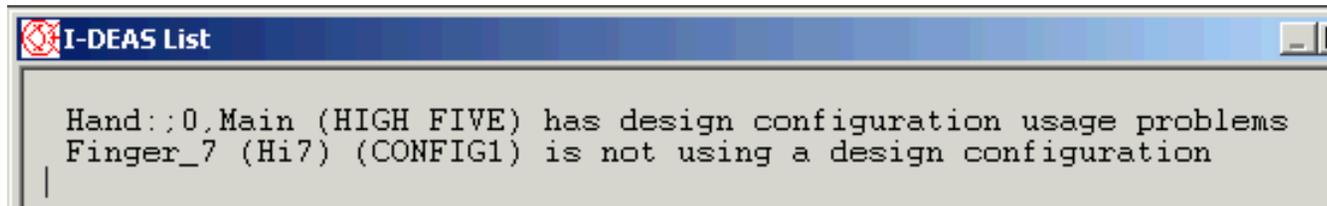
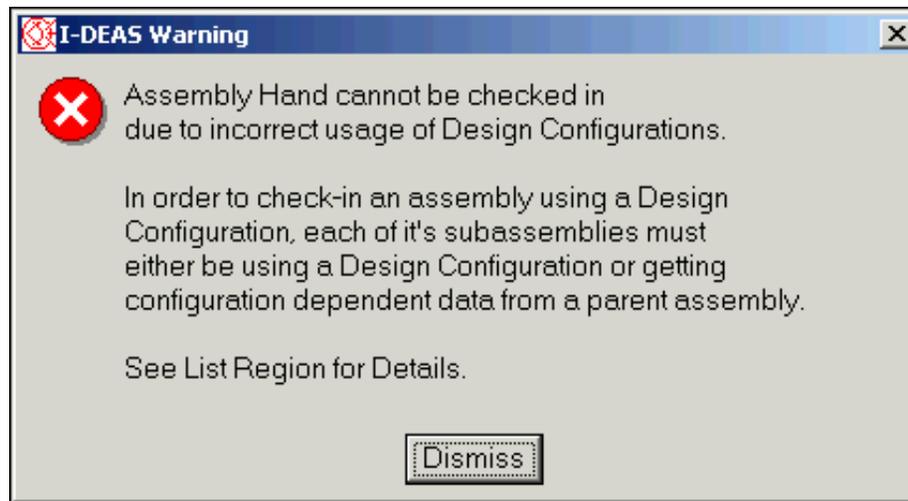


Assembly	Uses	Configuration
:Hand;	-->	HIGH FIVE (DESIGN)
:Finger;1	-->	STRAIGHT (DESIGN)
:Finger;1...		from Hand (HIGH FIVE)
:Finger;1...	-->	STRAIGHT (DESIGN)
:Finger;1...	-->	STRAIGHT (DESIGN)
:Thumb;...	-->	STRAIGHT (DESIGN)

Unused

Check-In Errors – 1

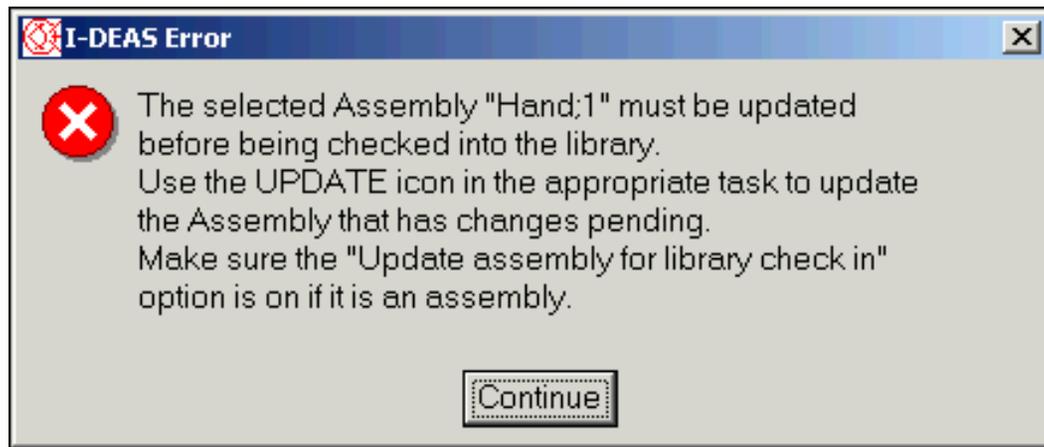
Sometimes I-DEAS warning messages are handy and informative. The warning explains the problem, the list window provides useful information, and it should be pretty clear what needs to happen to fix the problem



In the event it isn't clear... in this case, the parent assembly is tagged as a Design Configuration, but one of the subassemblies is not. That sub must either also be tagged, or must be unused.

Check-In Errors – 2

Sometimes I-DEAS warning messages are less than informative. The warning explains the problem, sort of, but may not provide enough information to indicate a possible solution. Worse, the List Window may not have any additional information.

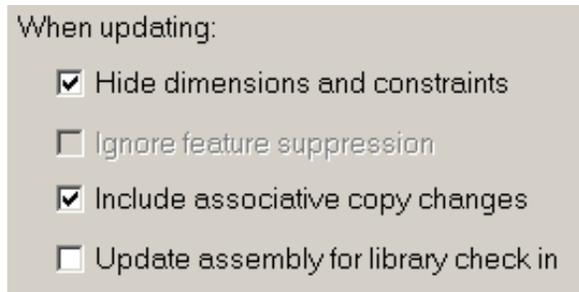
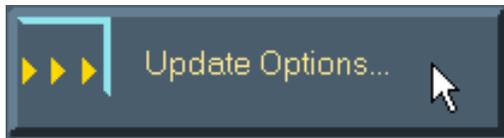


Worse, the message in question makes a questionable recommendation.

In this case, a change of some kind was made that affected a configuration that was tagged as a Design Configuration. However, that configuration was not activated or updated before attempting check in. Unfortunately, I-DEAS does not provide many hints regarding which configuration needs to be updated, and there could be more than one. In the event of such a change, all Design Configurations must be activated (Used) and updated prior to check in.

Check-In Errors – Workaround

The previous error message referred to an option called “Update assembly for library check in.” This is found under **Update Options**.



With this option turned on, I-DEAS will automatically – behind the scenes – update all configurations when **Update** is hit. This will naturally take longer than a regular update, but can be significantly faster than individually activating each configuration. Plus, the check in should succeed afterwards. For those reasons, it tends to be popular.

BE WARNED, however, that turning this toggle on essentially defeats the primary reason for having Design Configurations in the first place. By requiring that a Design Configuration be activated before check in, the assumption is that at least some form of visual inspection of the configuration will happen, since it is right there on the screen. The user can verify that the impact of any changes was handled correctly. However, since this option does the update behind the scenes, the Design Configurations are never visually verified. The check in proceeds as if nothing is wrong, and the user is none the wiser if there are indeed problems. What’s worse, any downstream users will assume that the configuration is correct, since it is tagged as a Design Configuration, and will make design assumptions on incorrect data. Design Configurations were specifically created to avoid precisely this scenario, so use this toggle sparingly and cautiously.

The End

Questions? Comments?

Brian Slick

bslick@ferno.com