



# **3D Visualization (3DV)**

**Release 35 (14.0.8)**

Last Modified: 04/17/2026

# Copyright Information

Copyright © 2026 Aras Corporation. All Rights Reserved.

Aras Corporation  
100 Brickstone Square  
Suite 100  
Andover, MA 01810  
Phone: 978-691-8900

## Notice of Rights

Copyright © 2026 by Aras Corporation and/or its affiliates. All rights reserved.

This document is protected by U.S. and international copyright laws and conventions. No copyright may be obscured or removed from this document. This document may not be modified or altered, or reproduced or transmitted in any form, without the explicit permission of the copyright holder.

Aras Innovator, Aras, and the Aras Corp "A" logo are registered trademarks of Aras Corporation in the United States and other countries.

All other trademarks referenced herein are the property of their respective owners.

## Notice of Liability

This document is provided for informational purposes only, and the contents hereof are subject to change without notice. The information contained in this document is distributed on an "as is" basis, without warranty of any kind, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or a warranty of non-infringement. Aras shall have no liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this document or by the software or hardware products described herein.

# Table of contents

<b>Administration Guide</b>	<b>8</b>
Introduction	9
Overview	10
Enabling Control of What Content is Displayed	11
Enabling Control of How Content is Displayed	12
3D Visualization Navigation Tool	13
Known Limitations	14
<b>Installation and Configuration</b>	<b>15</b>
Dynamic Viewer Installation Configuration	17
Out of the Box Setup	18
Customization Options	19
Streaming Viewer Configuration	20
Out of the Box Setup	21
Customization Options	22
<b>Aras 3D Viewers</b>	<b>23</b>
Streaming Viewer	30
CAD Data Model	31
Streaming Viewer Process Overview	34
Streaming Viewer Rendering Types	36
Streaming Viewer vs Dynamic Viewer	38
Tree Grid View vs. Model Browser	40
View Function Comparison	43
Visual Collaboration for Aras 3DV	44
View Modes	45
Saved Views	46
Modifying Default Saved Views Label	48
Digital Mockup	53
Context Menu Functions	54
Assembly Level Geometry Support for NX Files	55



<b>Dynamic Enabling</b>	<b>59</b>
Dynamic Enable Process	61
Enabling Legacy CAD Items to Work with Dynamic Visualization	63
<b>Creating Query and Tree Grid View Definitions</b>	<b>65</b>
<b>Query Definitions</b>	<b>66</b>
CAD / CAD Structure Data Model	67
Base Query Definition	69
Customizing the Query Definition	72
<b>Tree Grid View Definitions</b>	<b>85</b>
Default Tree Grid View Definition	86
Customizing the Tree Grid View Definition	90
Tree Grid View and 3D View Synchronization	95
Selecting a Dynamic View Definition	98
Auto-Execution	100
Enabling the Open Context Menu Item	101
<b>Alternate Query Processing</b>	<b>112</b>
<b>Overview</b>	<b>113</b>
<b>Implementing a Query Processor</b>	<b>115</b>
Create the Query Definition	116
Create the Tree Grid View	117
Create the Dynamic View Definition with Data Processor Method	118
Create the Data Processor Method	120
Create the Query Processor DLL	124
Processing Combined Rows	134
Rendering Configurations	135
<b>Deploying a Query Processor</b>	<b>138</b>
Build and deploy the DLL	139
Define the Data Processor Method	140
Create the Query/Tree Grid View/Dynamic View Definitions	141
Help files	142
<b>DynamicView Client-Side API</b>	<b>143</b>
Accessing DynamicView	144
dynamicView.addModel(itemIds)	145
dynamicView.removeModel()	146



dynamicView.isolateSelectedNodes()	147
dynamicView.setVisibilitySelectedNodes(isVisible)	148
dynamicView.hideAllOtherNodes()	149
dynamicView.fitAllNodes()	150
dynamicView.resetView()	151
dynamicView.displayAllNodes()	152
dynamicView.updateContextMenu()	153
<b>Using JT/STEP Converter</b>	<b>154</b>
<b>Appendix</b>	<b>156</b>
Customization Example with DynamicView API	179
Writing a Custom JS Function	180
Creating a New TGV Context Menu Item	185
TGV Context Menu Customization Results	189
Reverse Proxy Server	191
<b>Installation Guide</b>	<b>201</b>
Introduction	202
Scope	203
Target Audience	204
Aras 3DV Architecture Overview	205
Software and Hardware Requirements	206
Aras CAD Converter Requirements	207
Aras 3DV Licenses	209
CAD Converter License Activation	210
Download Aras 3DV	212
Aras 3D Visualization Installation	213
Aras Innovator Code Tree	214
Notification and Backup	216
Component Installation Order	218
Installation of Dynamic 3D Viewer After Using Monolithic 3D Viewer	219
Installation of Streaming 3D Viewer After Using Monolithic 3D Viewer or Dynamic Viewer	220

Automated Aras 3DV Installation	221
Automated Aras 3DV Installation for Streaming Viewer	229
Manual Aras 3DV Installation	241
Confirming Aras 3DV Installation	269
<b>Troubleshooting</b>	<b>270</b>
Unauthorized Access Exception	271
500.19 Internal Server Error	272
404: Not Found	273
3D PDF Is Not Activated in Firefox	275
<b>Upgrading from Version 14+</b>	<b>276</b>
<b>Introduction</b>	<b>277</b>
Scope	278
Target Audience	279
<b>Prerequisites</b>	<b>280</b>
Requesting a Feature License	281
Installing the License	282
Eligibility Verification	284
<b>Aras 3D Visualization Download</b>	<b>285</b>
<b>Upgrade Overview</b>	<b>286</b>
Upgrade Tools	287
Aras 3D Visualization Components and Modules	288
Upgrade Scenarios	289
Installing Dynamic 3D Viewer During Upgrade	290
Installing Streaming 3D Viewer During Upgrade	291
<b>Pre-Upgrade Procedures</b>	<b>292</b>
Aras Innovator Code Tree	293
Notification and Backup	295
<b>Automated Upgrade</b>	<b>296</b>
<b>Post-Upgrade Code Tree Cleanup Operations</b>	<b>305</b>
Conversion Server File Cleanup	306

Client Library File Cleanup	307
Post-Upgrade Modification for the Streaming Viewer	308
Successful Upgrade Confirmation	309
<b>User Guide</b>	<b>310</b>
Introduction	311
Scope	312
Target Audience	313
Introduction to Aras 3D Visualization (3DV)	314
CAD Documents and 3DV	315
Managing 3D CAD Files	317
Accessing 3DV Functionality	320
Monolithic vs. Dynamic Viewer	322
Dynamic vs. Streaming Viewer	323
Aras 3DV UI	324
3D Scene	325
Aras 3DV Toolbar	326
Viewing Toolbars	327
Markup Toolbars	329
Aras 3DV Context Menus	331
3D Scene Context Menu	332
Part Context Menu	333
General 3DV functionality	335
Viewing CAD Models	336
Zooming	339
3D Viewer Preferences	345
Configuring Display Style	350
Measuring CAD Models	355
Exploding CAD Models	360
Cross Section Viewing of the CAD Model	362
Exporting SVG Image File	372
Marking up CAD Models	374
Drawing on CAD Models	375



Highlighting Areas on CAD Models	378
Putting Labels on CAD Models	382
Selecting Markups on CAD Models	389
Deleting Markups on CAD Models	392
Saving and Sharing Markups	393
<b>Monolithic Viewer</b>	<b>395</b>
Product Manufacturing Information	396
Model Browser in Monolithic Viewer	400
Browsing CAD Model Assemblies and Parts with Monolithic Viewer	403
Browsing CAD Model Views with Monolithic Viewer	407
<b>Dynamic and Streaming Viewer</b>	<b>410</b>
Dynamic Viewer and Streaming Viewer Context Menus	413
3D Scene Context Menu in Dynamic Viewer and Streaming Viewer	414
Part Context Menu in Dynamic and Streaming Viewers	415
Digital Mockup	416
Markup Lines in Dynamic and Streaming Viewers	417
Adding Markup Lines by Intersection	418
Adding Markup Lines by Center	421
Removing Markup Lines	425
Manual Geometry Transformation	429
Model Browser in Dynamic and Streaming Viewers	430
Saved Views	432
Multi-Selection	433



# Administration Guide



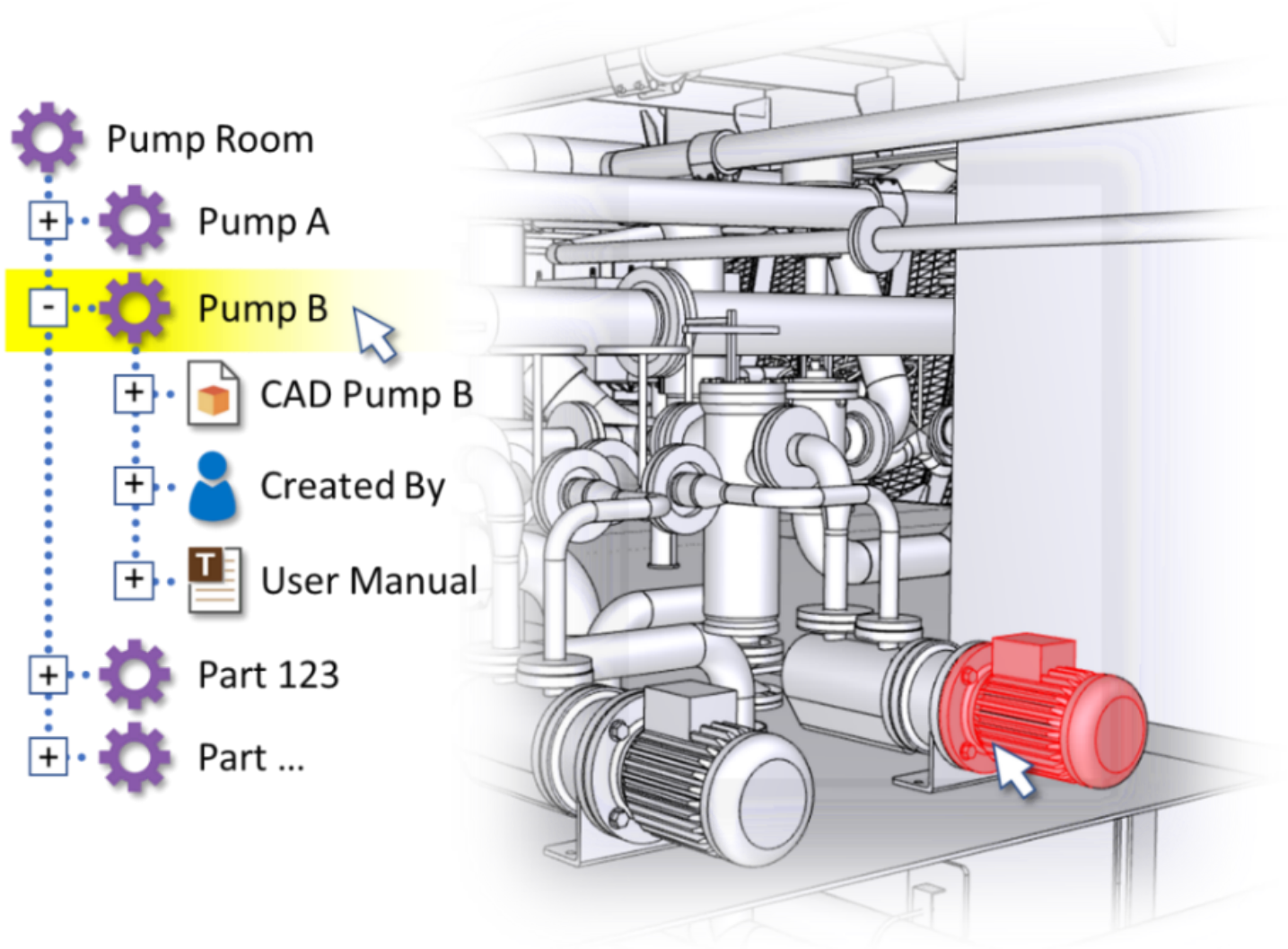
## Introduction

This Administration Guide provides instructions for configuring Aras 3D Visualization (3DV) and all its features.



## Overview

Aras Innovator is a framework, and 3D Visualization is a tool within it. The goal is to be able to apply or use 3D Visualization throughout the various phases of a Product's lifecycle. The approach is to provide a level of configurability that enables end-user customizations so that 3D Visualization can be applied based on a customer's business need. These enhancements include the ability to identify what 3D content is displayed, how it is rendered, and what related content is included. Collectively these enhancements are known as Dynamic Product Navigation (DPN).



## Enabling Control of What Content is Displayed

Aras Innovator has Query Builder and Tree Grid View, which provide an end-user with a graphic editor to define an ItemType query and bind the elements of that query to a Tree Grid View. The concept is like a Model and View respectively in the Model-View-Control paradigm typically used as a design guide in software engineering. The vision was to use these queries—known as Query Definitions—as a reusable data structure in such a way that other types of views could be constructed from them, like Graph Navigation Views. Query Definitions and their included conditional logic are used with DPN to identify the elements to be rendered in a 3D View, thus providing a convenient mechanism for users to control what elements are shown in the 3D Viewer.



## Enabling Control of How Content is Displayed

Tree Grid View and Dynamic View Definitions describe how to visualize the results when a Query Definition is executed. Tree Grid View Definitions map elements of a Query Definition to Tree Nodes and columns. These elements are the components that define a Tree Grid View. Thus, a specific Query Definition is executed to identify Query Results which are then visualized as a Tree Grid View based on the mappings in a Tree Grid View Definition. DPN uses both Query Definitions and Tree Grid Views. Views can provide additional visual information simply based on color and transparency. That is, render the 3D geometry using a particular color to emphasize (or de-emphasize) when viewing with other parts. Likewise, the level of transparency can be used for a similar effect. Dynamic View Definitions allow users to identify a Data Processor which uses a Query Processing API to implement customization points that allow end-users to adjust or define 3D color and transparency used in a 3D View.



## 3D Visualization Navigation Tool

With the Dynamic Product Navigation (DPN):

- 3D Views are generated dynamically as the product of a defined Query.
- Related content can be visualized together with 3D components.

3D Visualization can be used as a navigation tool whereby users select 3D components in a view and see related Business Objects (related Items) with the ability to open these Items directly from the 3D View.

## Known Limitations

Out of the box, Aras Innovator uses the CAD and CAD Structure relationship ItemTypes to capture the Bill of Materials (BOM) information for a mechanical Assembly. Part instance and transformation data are required for Dynamic Visualization, and this information exists in **CAD Instance** Items. The software logic in the CAD Converter creates CAD Instance Items with the appropriate transformation information. The support of alternate data models exists as the ability to implement a custom Query Processor.



## Installation and Configuration

### CAD Conversion

#### Install CAD Converter

To set up CAD Conversion, refer to the *Aras 3D Visualization 35 – Installation Guide*, available in the **Documentation** folder of the 3D Visualization CD Image, which subscribers can obtain from the **Aras 3D Visualization** folder on the Aras FTP site.

#### Use Legacy View Files

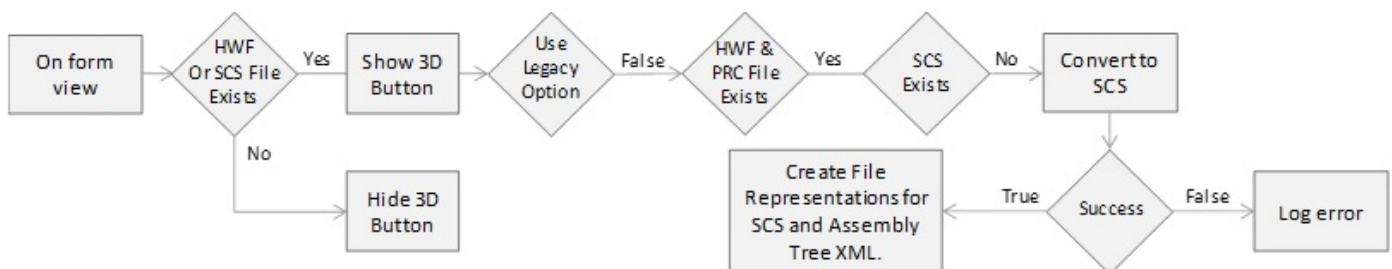
Aras Innovator provides backward compatibility in both the data model and viewing features. With the current Viewer, users can convert legacy HWF view files to the new Stream Cache Single (SCS) file format using an automated asynchronous process. The **CAD** Items can either use the HWF, SCS or SCZ format.

#### Preference Settings

Users can choose to always display legacy HWF files instead of converting them to SCS by selecting the **Use Legacy 3D View Files** checkbox on the **Secure Social** tab of a given Preference Item.

#### Convert Legacy View Files

The conversion process for legacy view files begins when a user asks to view a **CAD Document** or **Part** Item with a related **CAD Document** Item that includes a legacy view file (HWF). The following figure describes the process.



The existence of an HWF, SCS file determines whether the 3D Viewer can be opened. If 3D Viewer opens up, a button appears on the sidebar enabling users to open the Viewer. If either of these files does not exist, the user is unable to open the 3D Viewer. The **Use Legacy 3D View Files** setting value of the given Preference Item and the existence of an SCS file determines whether it is necessary to convert the file. If the following conditions exist, an ad hoc, asynchronous conversion process starts:

- The Use Legacy 3D View Files Preference setting is False.
- HWF and PRC files exist.
- An SCS file has not been created previously.

A successful conversion process results in a new file representation that points to the generated SCS and Assembly Tree files. Conversion process errors should be logged. Subsequent attempts to open the same CAD document after a successful conversion result in viewing the generated SCS file.

### **CAD Unit Support**

3DV introduces a new Extended Property to FileRepresentation type called 'xp-unit-scale'

In the conversion process, the Unit attributes are searched in the root node of the conversion XML and set the value to XML File Representation 'xp-unit-scale'.

In generating ModelXM, 'xp-unit-scale' is added to the value from the Root XML File Representation (from the Root CAD native file) to generate ModelXML. The ModelXML then goes to HOOPS Viewer to visualize the generated structure.

#### **Important**

Unit scale in this case helps to visualize assemblies with incorrect default measurements. Correct display and correct measurements



## Dynamic Viewer Installation Configuration

The installation process creates the default Dynamic Viewer configuration. For installation details, refer to the *Aras 3D Visualization 35 – Installation Guide*.

The following section explains the command arguments. Any outputs not required by a given implementation can be removed from the command arguments.



## Out of the Box Setup

When Aras 3D Visualization is installed, the following OOTB setup is used in the **ConversionServerConfig.xml** file:

```
<AssemblyCommand dynamicEnabled="True" arguments="--  
sc compute bounding boxes 'All' --input pdf template file 'C:\HOOPS  
Converter\templates\Blank Template L.pdf' --output pdf  
'%filepath%\%filename%.pdf' --output png '%filepath%\%filename%.png' --  
output png resolution '150x150' --output scs '%filepath%\%filename%.scs' -  
-output xml assemblytree '%filepath%\%filename%.xml' --output prc  
'%filepath%\%filename%.prc' --background color '1.0, 1.0, 1.0' --  
output_logfile '%filepath%\%filename%.log'" />
```

### Important

Due to differences between the Windows and Linux file systems, it is required to use OS-specific path separators in paths. Because the Linux file system is case-sensitive, there is a significant difference between these file names: **./path/to/file.xml** and **./path/to/File.xml**. For more information about cross-platform development, refer to the Cross-platform development section in the Aras Innovator 31 - Programmer's Guide.



## Customization Options

The following table describes the command arguments. Any outputs not required by a given implementation can be removed from the command arguments.

Command-line Argument	Required	Description
-- sc_compute_bounding_boxes	Yes	Positions the camera in the Viewer and prioritizes the rendering order of an assembly. The bounding box information will be extracted from a native CAD file.
--input_pdf_template_file	No	Required only if generating viewable PDF files for an Item.
--output_pdf	No	Required only if generating viewable PDF files for an Item.
--output_png	Yes	Produces a thumbnail image for an Item.
--output_scs	Yes	Enables the Monolithic and Dynamic Viewers.
--output_xml_assemblytree	Yes	Maps 3D component geometry.
--output_prc	No	Required only if generating PRC files for an Item that may be used for industry standard archival purposes.
--background_color	Yes	Sets the background color for thumbnail images (png) and the 3D PDF. The default is black. However, a background color other than white may affect the display of the image in the Item Form.
--output_logfile	Yes	Sets a name of a log file where the HOOPS Converter will write error and warning messages.
--drawings_mode	No	Controls what to import from drawing files: 0: only 3D 1: only drawings 2: both 3D and drawings The default is 1.

Please see the included documentation for a complete description of available parameters and their functions in the HOOPS Communicator Documentation 2025.4.0.



## Streaming Viewer Configuration

The installation process creates the default Streaming Viewer configuration. For installation details, refer to the *Aras 3D Visualization 35 – Installation Guide*.

The following section explains the command arguments. Any outputs not required by a given implementation can be removed from the command arguments.



## Out of the Box Setup

When Aras 3D Visualization is installed, the following OOTB setup is used in the **ConversionServerConfig.xml** file:

```
<AssemblyCommand arguments="--sc compute bounding_boxes 'All' --  
input pdf template file 'C:\Aras\14SP10\HOOPS  
Converter\Templates\Blank Template L.pdf' --output pdf  
'%filepath%\%filename%.pdf' --output png '%filepath%\%filename%.png' --  
output png resolution '150x150' --output xml assemblytree  
'%filepath%\%filename%.xml' --output prc '%filepath%\%filename%.prc' --  
background color '1.0, 1.0, 1.0' --output log file  
'%filepath%\%filename%' streamingEnabled="True"/>
```

### Important

Due to differences between the Windows and Linux file systems, it is required to use OS-specific path separators in paths. Because the Linux file system is case-sensitive, there is a significant difference between these file names: **./path/to/file.xml** and **./path/to/File.xml**. For more information about cross-platform development, refer to the Cross-platform development section in the Aras Innovator 31 - Programmer's Guide.

### Important

The Streaming Viewer currently cannot be deployed in a cloud environment.

The HOOPS Server must be deployed with networked file access to a single vault containing view files for rendering.

### Important

Only one Streaming Viewer can be installed on one machine at a time.



## Customization Options

The following table describes the command arguments. Any outputs not required by a given implementation can be removed from the command arguments.

Command-line Argument	Required	Description
-		Positions the camera in the Viewer and prioritizes the rendering order of an assembly. The bounding box information will be extracted from a native CAD file.
sc_compute_bounding_boxes	Yes	
--input_pdf_template_file	No	Required only if generating viewable PDF files for an Item.
--output_pdf	No	Required only if generating viewable PDF files for an Item.
--output_sc	Yes	Full path of SC Model to generate. Warning: This option will recursively delete all existing files in the target directory.
--sc_create_scz	Yes	If set, then generated SC models will be .scz files. Default: False
--sc_compress_scz	No	If set, then generated .scz files will be compressed. Default: True
--output_png	Yes	Produces a thumbnail image for an Item.
--output_xml_assemblytree	Yes	Maps 3D component geometry.
--output_prc	No	Required only if generating PRC files for an Item that may be used for industry standard archival purposes.
--background_color	Yes	Sets the background color for thumbnail images (png) and the 3D PDF. The default is black. However, a background color other than white may affect the display of the image in the Item Form.
--output_logfile	Yes	Sets a name of a log file where the HOOPS Converter will write error and warning messages.

Please see the included documentation for a complete description of available parameters and their functions in the HOOPS Communicator Documentation 2025.4.0.



## Aras 3D Viewers

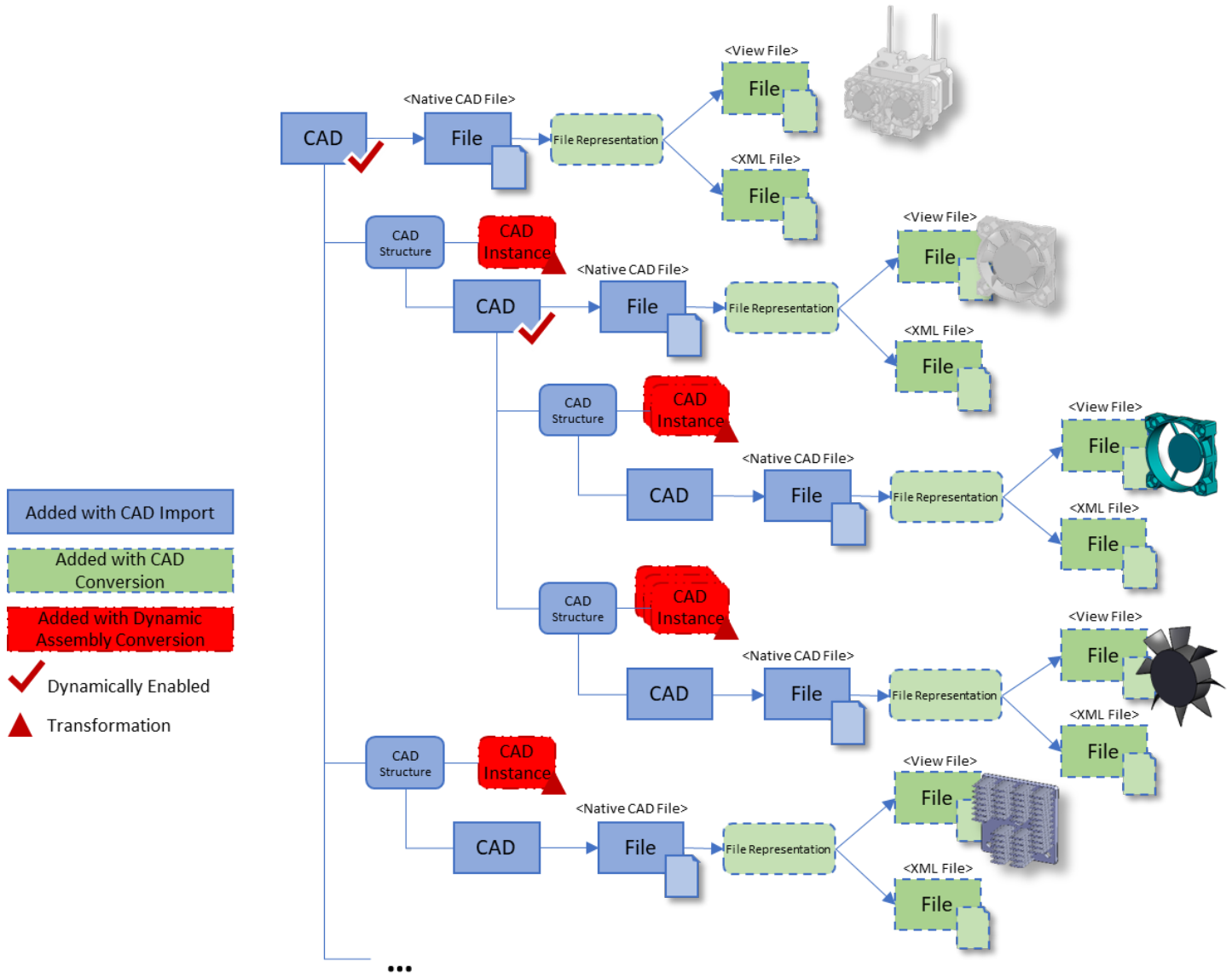
### Dynamic Viewer

This section describes the Dynamic Viewer and compares its features with the Monolithic Viewer. It also provides an overview of the assumed ItemType data model, from which the data required by Dynamic Viewer is queried.

#### CAD Data Model

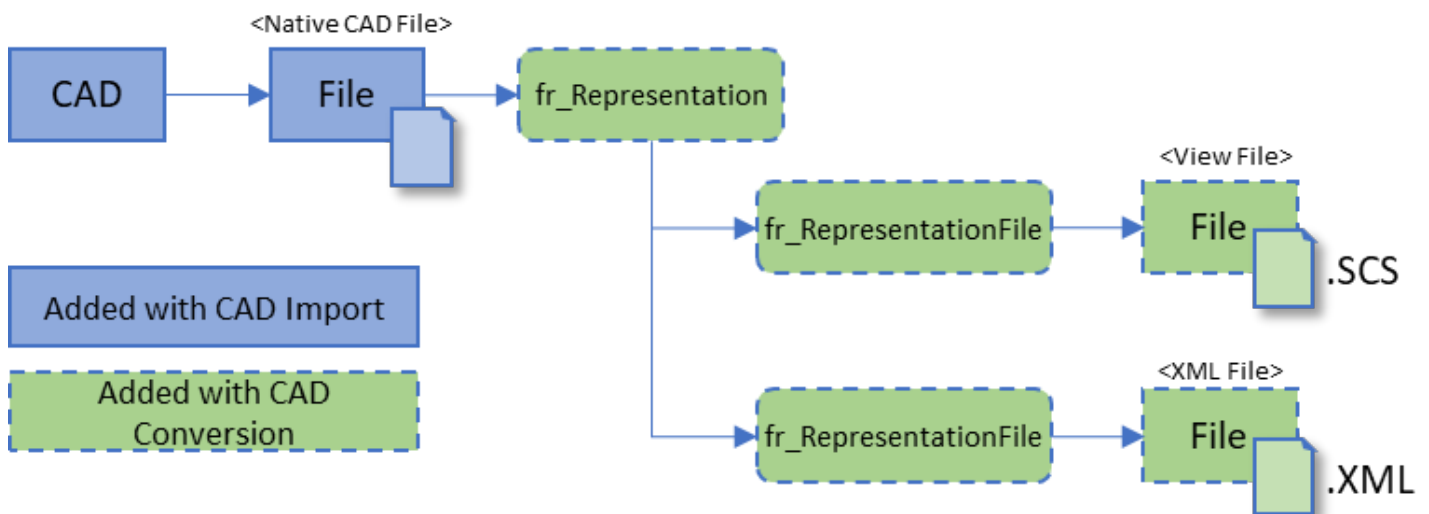
Before reviewing the Dynamic Viewer features, it is important to understand the **CAD** ItemType data model, so it is clear what data is required to support Dynamic Visualization and where this information is stored and retrieved by default. This section describes the process of creating **CAD** Items, their related ItemTypes, and **CADFile** conversions using the following diagram.





The **CAD** Items and **CAD Structure** Relationship Items are typically created by a CAD Connector when CAD native files are checked into the Aras Innovator. A **CAD** Item represents a single 3D CAD assembly, part, or 2D drawing. For this document, **CAD** Items representing 3D CAD assemblies and parts are assumed. In this case, the native CAD file is attached (via the **File** ItemType property) to each **CAD** Item. If this **CAD** Item represents an assembly, all related subassemblies and parts will be included in separate **CAD** Items and referenced via **CAD Structure** Items. The **CAD** and **CAD Structure** Items represent a mechanical BOM.

CAD Conversion is the process of generating alternate versions of native CAD data. It is triggered by the existence of a CAD file. <sup>[1]</sup> The Monolithic and Dynamic Viewers render 3D component geometry stored in a Stream Cache Single (SCS) view file. Note that thumbnail images, PRC, 3D PDF, and other converted formats can also be created as part of the CAD Conversion process and stored in a **CAD** Item or its related Items. An XML file is also created and stored. This XML file is generated from the conversion process and used to identify all the instances of parts and subassemblies included in a CAD assembly. This information is used to map 3D component geometry in the Monolithic Viewer and generate CAD Instances and 3D Transformations for the Dynamic Viewer. The SCS view and XML files are stored as related Items attached to a **File** Item for the native file. Note that the figure below uses a single Relationship graphic labeled **File Representation** to simplify the diagram. However, File Representations use two RelationshipTypes.

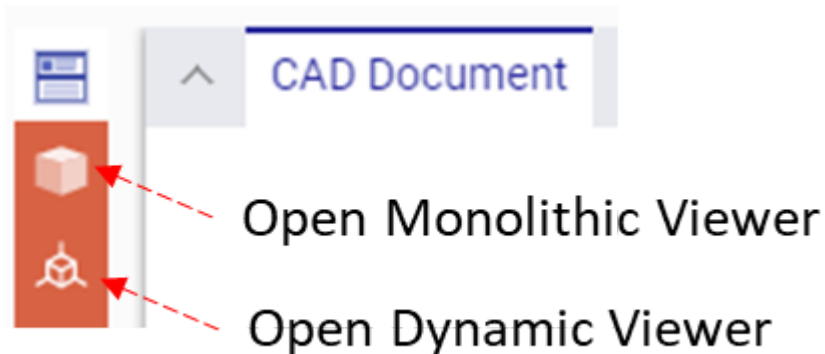


For the Dynamic Viewer, the conversion process appends the **CAD Instance** Items for all instances of an associated (related) **CAD** Item. **CAD Instances** contain a property that stores the 3D Transformation information as a 4X4 matrix. In addition, the added **Dynamic Enabled** Boolean property is set to **true** for a **CAD** Item with an assembly, when all data necessary to render that CAD assembly using the Dynamic Viewer has been created. Note that these Instances and their transformation information are created automatically by the 3D Conversion Process based on information extracted from a native CAD assembly file.

### Monolithic vs Dynamic Viewer

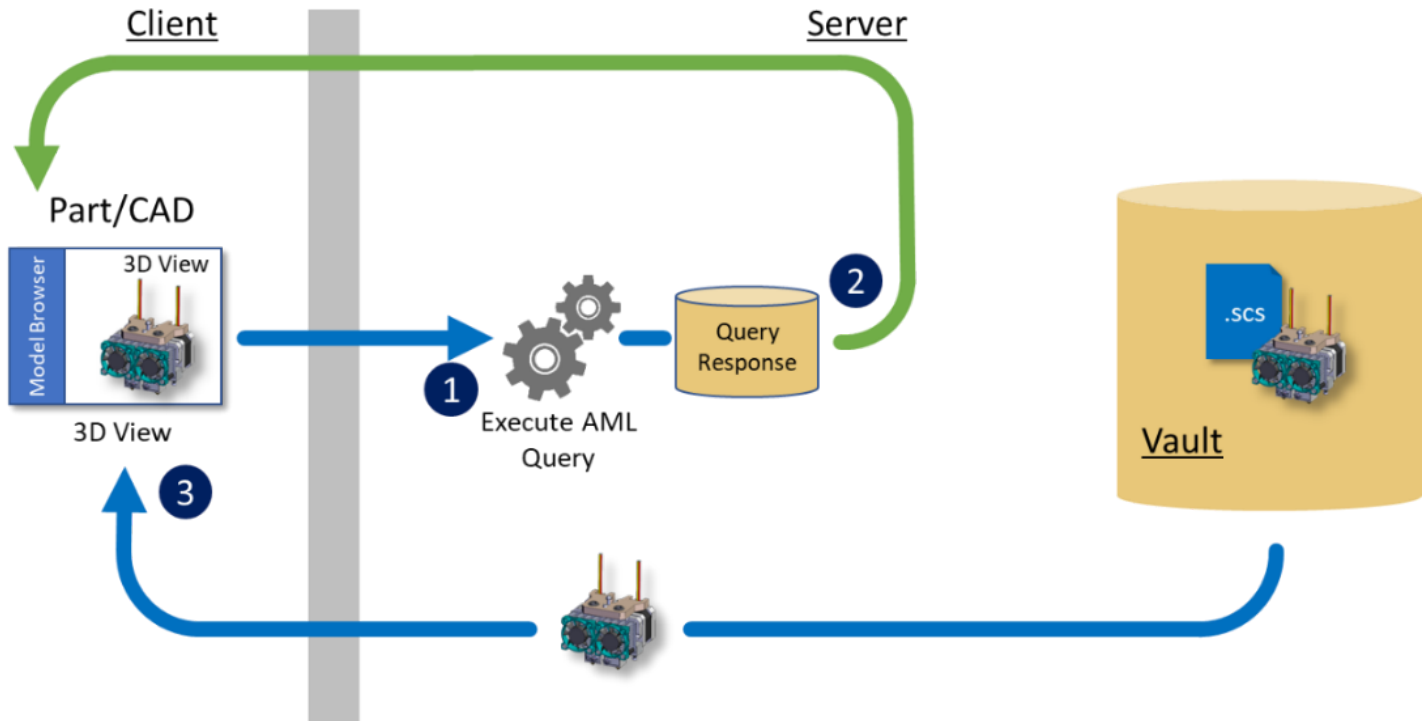


This section compares the existing features of the Monolithic and Dynamic Viewers. The below figure shows the buttons on the sidebar of a **CAD** Item form that opens the Viewers.



### Monolithic Viewer Process Overview

The Monolithic Viewer displays a single SCS view file attached to a corresponding **CAD** Item. If it is an assembly view file, the monolithic view file includes the 3D component geometry for all subassemblies and parts included in this assembly at the time the CAD file was checked in. In essence, it is a static view of the assembly because it will not include any geometry changes to subcomponents made after the assembly view file was created. The following figure shows the process of rendering the view file in the Monolithic Viewer.

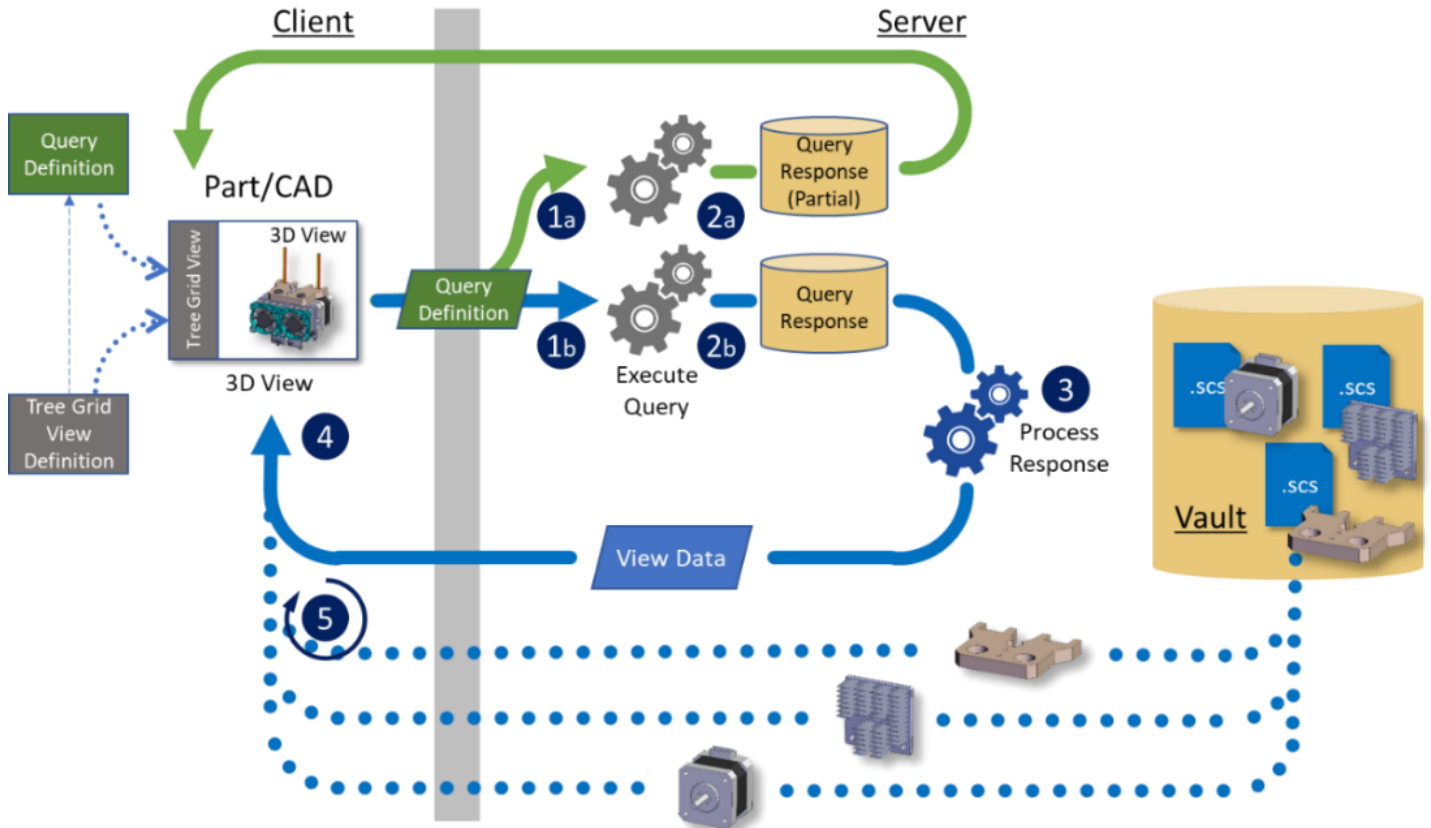


When the Monolithic Viewer is opened, a static AML query is executed to retrieve the multi-level CAD structure of an opened **CAD** Item. The results are displayed in a tree user interface called the simple Model Browser of the Viewer. A request is made to load a single SCS view file associated with the opened **CAD** Item, the 3D component geometry of which is displayed in the Viewer. The XML data generated from the conversion process is also retrieved, and the information contained within it is used to map the Items displayed in the simple Model Browser with the associated 3D components in the 3D View scene. This provides selection synchronization between the Model Browser and the 3D View.

### Dynamic Viewer Process Overview

The Dynamic Viewer is used only for assemblies and displays SCS view files as determined by the results returned from the execution of an associated Query Definition (QD). Unlike a monolithic view file for an assembly, the Query results for a dynamic Query target the view files for assembly parts and the instance and transformation data required to properly position each file given the returned assembly hierarchy. The following figure shows the process of rendering the view file in the Dynamic Viewer.





The Dynamic Viewer consists of two main components: Tree Grid View (TGV) and the 3D View. The TGV displays the results of the associated QD as defined by the chosen Tree Grid View Definition. The Tree Grid View is a composite of a Tree View and a Table (or Grid). The left-most column contains a hierarchical Tree View showing all the related contents starting from (rooted by) the **CAD** Item from which the Dynamic View was opened.

Upon refreshing the View, the system executes the associated QD in two simultaneous operations. The first executes the Query to populate the TGV. The second executes the Query and processes the complete response.

The TGV uses the partial response based on the configuration of the associated TGV Definition.<sup>[2]</sup> By default, the TGV is lazy-loaded: only a portion is returned to a client and displayed. The full response is processed to generate the view data so that all component parts are identified.



While processing the Query results for the 3D View, XML data is constructed that identifies the assemblies, parts, part instances, and their transformations for each. Parts will have links to the respective view files in the Aras Innovator Vault.

The 3D Viewer processes the view data (XML) and sends subsequent requests to the Aras Innovator Vault to retrieve each of the corresponding view files to render. The 3D Viewer processes and renders the view files individually and in sequence.



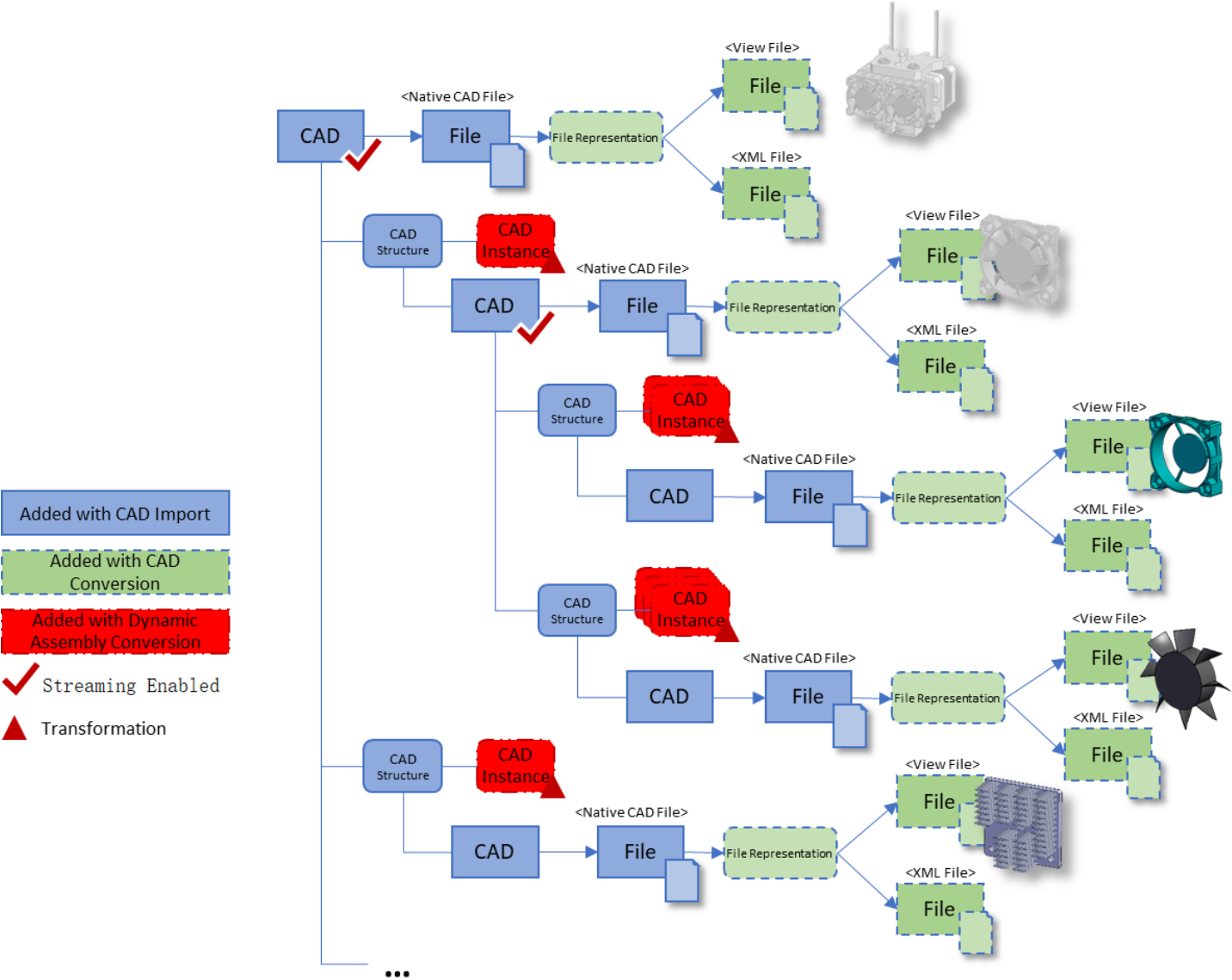
## Streaming Viewer

This section describes the Streaming Viewer and compares its features with the Dynamic Viewer. Streaming Viewer has the same functionality as the Dynamic Viewer with improved rendering speed. Streaming Viewer requires a separate Windows Service called Hoops Server.



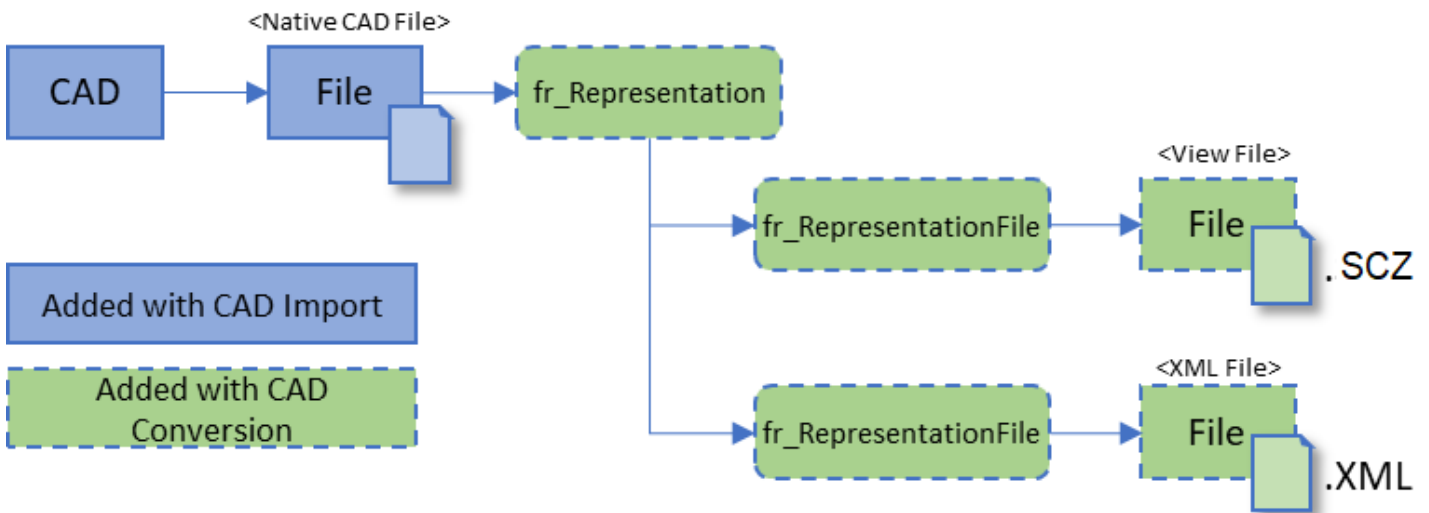
CAD Data Model

Before reviewing the Streaming Viewer features, it is important to understand the CAD ItemType data model, so it is clear what data is required to support Dynamic Visualization and where this information is stored and retrieved by default. This section describes the process of creating CAD Items, their related ItemTypes, and CADFile conversions using the following diagram.



The CAD Items and CAD Structure Relationship Items are typically created by a CAD Connector when CAD native files are checked into the Aras Innovator. A CAD Item represents a single 3D CAD assembly, part, or 2D drawing. For this document, CAD Items representing 3D CAD assemblies and parts are assumed. In this case, the native CAD file is attached (via the File ItemType property) to each CAD Item. If this CAD Item represents an assembly, all related subassemblies and parts will be included in separate CAD Items and referenced via CAD Structure Items. The CAD and CAD Structure Items represent a mechanical BOM.

CAD Conversion is the process of generating alternate versions of native CAD data. It is triggered by the existence of a CAD file. The Streaming Viewer Viewers render 3D component geometry stored in a Compressed Stream Cache (SCZ) view file. Note that thumbnail images, PRC, 3D PDF, and other converted formats can also be created as part of the CAD Conversion process and stored in a CAD Item or its related Items. An XML file is also created and stored. This XML file is generated from the conversion process and used to identify all the instances of parts and subassemblies included in a CAD assembly. This information is used to generate CAD Instances and 3D Transformations for the Streaming Viewer. The SCZ view and XML files are stored as related Items attached to a File Item for the native file. Note that the figure uses a single Relationship graphic labeled File Representation to simplify the diagram. However, File Representations use two RelationshipTypes, as shown in the following figure.



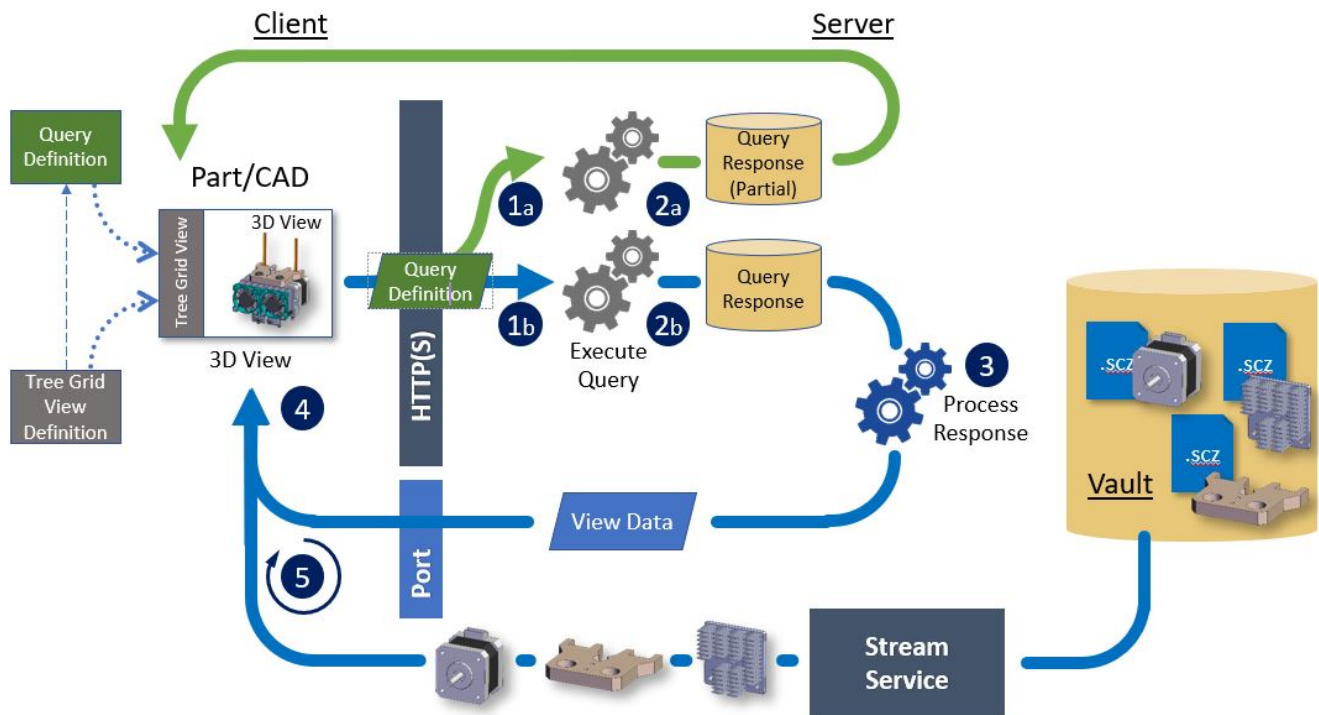
For the Streaming Viewer, the conversion process appends the CAD Instance Items for all instances of an associated (related) CAD Item. CAD Instances contain a property that stores the 3D Transformation information as a 4X4 matrix. In addition, the added Streaming Enabled Boolean property is set to true. Note that these Instances and their transformation information are created automatically by the 3D Conversion Process based on information extracted from a native CAD assembly file.



### Streaming Viewer Process Overview

The Streaming Viewer is used only for assemblies and displays SCZ view files as determined by the results returned from the execution of an associated Query Definition (QD). Below figure shows the process of rendering the view file in the Streaming Viewer.

## Streaming Viewer Rendering Process (Shattered)



Similar to the Dynamic Viewer, the Streaming Viewer consists of two main components: Tree Grid View (TGV) and the 3D View. The TGV displays the results of the associated QD as defined by the chosen Tree Grid View Definition. The Tree Grid View is a composite of a Tree View and a Table (or Grid). The left-most column contains a hierarchical Tree View showing all the related contents starting from (rooted by) the CAD Item from which the Streaming View was opened.

Upon refreshing the View, the system executes the associated QD in two simultaneous operations. The first executes the Query to populate the TGV. The second executes the Query and processes the complete response.



The TGV uses the partial response based on the configuration of the associated TGV Definition. By default, the TGV is lazy-loaded, only a response portion is returned to a client and displayed.

The full response is processed to generate the view data using SCZ files.

While processing the Query results for the 3D View, XML data is constructed that identifies the assemblies, parts, part instances, in single response which has links to the respective view files in the Aras Innovator Vault. The Streaming Viewer is supported only with a single vault.

Streaming visualization relies on a server-side component – Stream Service, that analyzes the current camera position of the client viewer and optimizes which components of each SCZ file to ‘stream’ to the client to be subsequently rendered. The Streaming Viewer communicates with the Stream Service via a dedicated network port.

### Streaming Viewer Rendering Types

The Streaming Viewer Rendering Types can be configured using the HOOPS\_Viewer\_Renderer\_Type variable.

### Client-Side Rendering

In Client-Side rendering type, the 3D model is built on the client side. The Stream Cache Server streams the 3D model data to the Hoops Web Viewer enabling the client hardware to render the 3D graphics using WebGL technology.

To select the Client-Side Rendering Type, in the HOOPS\_Viewer\_Renderer\_Type variable, change the Value field to Client.

**X HOOPS\_Viewer\_Renderer\_Type**

Edit [Refresh] [Undo] [Share] [List] [Share]

Variable

Name HOOPS\_Viewer\_Renderer\_Type

Value **Client** Client-Side Rendering

Default Value Client

### Server-Side Rendering

When a Web Viewer is set up for server-side rendering, all rendering of the 3D model is performed by the Graphic Processor Unit (GPU) on the web server. As the user interacts with the 3D model, the server renders each frame and sends an image back to the client's web browser for display. This



minimizes hardware requirements for the client. The images are sent in real time, so it appears as if the rendering is being done in real time on the local machine.

To select the Server-Side Rendering Type, in the HOOPS\_Viewer\_Renderer\_Type variable, change the Value field to Server.

## X HOOPS\_Viewer\_Renderer\_Type



### ^ Variable

**Name** HOOPS\_Viewer\_Renderer\_Type

**Value** Server **Server-Side Rendering**

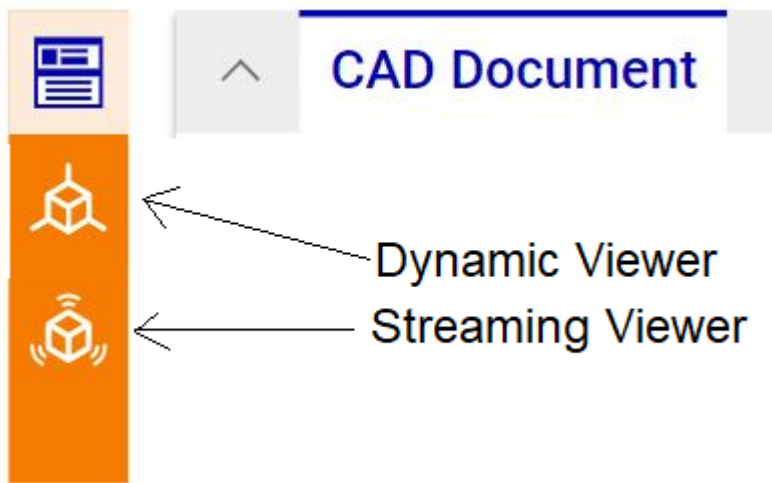
**Default Value** Client

#### Important

If the server where Streaming Viewer is installed has Graphic Processing Unit (GPU), then the `windowsServiceRespawnEnabled` parameter in the `Config.js` file should be set to "False".



## Streaming Viewer vs Dynamic Viewer



The Dynamic and Streaming Viewers have the similar functionalities. The difference between them is as follows:

- Dynamic Viewer supports SCS viewable files whereas the Streaming Viewer supports SCZ viewable files. Each is designed for a specific purpose.
- For Streaming Viewer, a separate window service called Hoops Server is used. The HOOPS Server initiates individual Stream Services to support each Streaming Viewer instance.
- Larger datasets can be streamed from a server to a client in a single request with the help of the Streaming Viewer unlike Dynamic Viewer which uses individual client/server HTTP requests.

### Important

Dynamic Visualization and Streaming Visualization each use different view files, and each are not compatible with the different Viewers.

Therefore, if the Streaming 3D Viewer is installed in an environment where the Monolithic or Dynamic 3D Viewers have been previously installed and used, all existing native files of existing CAD Documents need to be re-converted. There is no automated means to perform this reversion!

### Important

The Streaming Viewer currently cannot be deployed in a cloud environment.

The HOOPS Server must be deployed with networked file access to a single vault containing view files for rendering.

**Important**

Only one Streaming Viewer can be installed on one machine at a time.



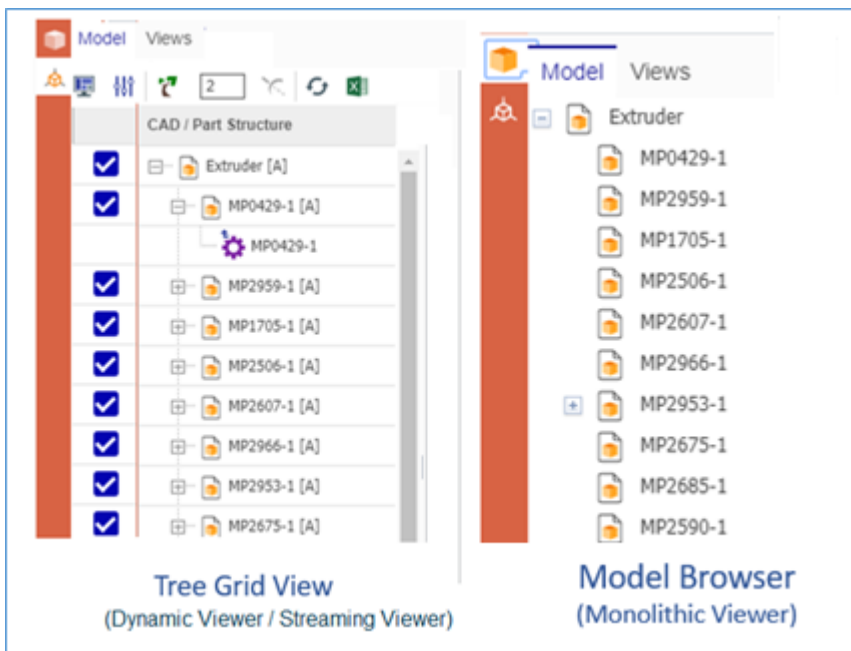
## Tree Grid View vs. Model Browser

The concept and vision of the Dynamic Product Navigation (DPN) require an alternate mechanism for displaying business objects (Items) associated with 3D components displayed in the 3D View:

- The *dynamic* part of DPN specifies that some conditional and customizable logic should determine 3D view content. A QD provides this ability.
- The *navigation* part of DPN specifies that the 3D View can be used to access related business objects (Items). A TGV Definition provides this ability.

The optimal choice for the Dynamic /Streaming Viewer is the TGV—a user interface component that displays Items and associated properties in a grid.

This section compares the features of the TGV Model Browser of the Dynamic Viewer as well as the Streaming Viewer against the simple Model Browser of the Monolithic Viewer to make it clear how each function and what information each provides.



## QD vs. AML

A TGV is populated with a response generated from the execution of an associated QD. In addition to the **CAD** Items that form the Query basis, administrators can add other related ItemTypes whose properties should be displayed in the TGV UI. Using TGV Definitions, administrators can choose



which Items to display, what Properties to include, how to format the Property values, static text, what columns to include, and alternative icons. For example, the figure shows the left-most Tree column with nodes for the following Items that display the following properties for a given Item:

- **CAD**: A **CAD** Item **Name** and revision in square brackets.
- **Part**: A Part Item Name.

In addition, two columns were added for the `is_released` and `is_current` Boolean properties that both are displayed as checkboxes.

In contrast, the simple Model Browser content is determined by the execution of a static (hard-coded) AML query on the CAD structure of an opened **CAD** or **Part** Item. Each node in the Tree uses the `keyed_name` Property of the **CAD** Item with the icon for the **CAD** ItemType. There are no other ways to configure the display other than the configuration of the `keyed_name` Property.

#### Partial vs. Full CAD Structure display

By default, a TGV is populated using only a portion of QD response results. When related content has multiple levels, a user determines which content to query based on selected nodes in the Tree column. Expanding a node re-executes the query from this node and adds new rows to the TGV. On the contrary, the simple Model Browser uses an AML to execute an exhaustive (fully recursive) search for the entire CAD BOM structure. The Tree is populated with the full list of CAD Items from the query.

Each approach has a trade-off. For the TGV, query execution is generally faster, much faster for large results. However, after the simple Model Browser is populated, all data exists on the client and can be readily viewed. This affects data synchronization between the simple or TGV Model Browser and 3D View. For the Dynamic Viewer, when a 3D component is selected in the 3D View, a directed set of sub-queries is performed to populate the TGV with all levels of the hierarchy up to the selected part component. Once the data is populated, the query should not be re-executed, but the initial population can take several seconds.

In general, and especially for large and deep assembly hierarchies, the lazy-loading approach used by the TGV will yield a better user experience and require less network data exchange and less memory for the client browser. Also, the number of peer nodes and depth of query responses is



configurable in the TGV Definition—administrators have control over the granularity of query response data.



## **View Function Comparison**

The Monolithic, Dynamic and Streaming 3D Viewers use the same technology— HOOPS Communicator. The main difference is how the 3d View is populated. However, there are also limitations associated with the available view functions that this section explains.

## **Product Manufacturing Information (PMI)**

The product management information (PMI) is typically stored at the assembly level within CAD native files. When dynamically generating assemblies in the Dynamic Viewer or Streaming Viewer, only view files of parts are included. Therefore, only PMI stored with a native file and included with the converted data in a view file can be viewed in the 3D View. PMI is not guaranteed to be positioned such that it does not interfere with other rendered 3D component geometry.

## **Configurations**

The Dynamic Viewer is only accessible for CAD assemblies. Like PMI, configuration data stored in an assembly view file is not accessible and likely not applicable when rendering an assembly based on some arbitrary query. Therefore, the Configurations interface is not included in the Dynamic Viewer.

The Streaming Viewer is accessible for both CAD assemblies and CAD Components.



## Visual Collaboration for Aras 3DV

The Aras 3DV is integrated with Visual Collaboration to help visualize complex product information using a simple UI. End users can freeze a 3D View scene, add 2D graphic and text markups, and save a resulting image with a comment in a discussion thread, which may be related to 3D design, for example. There is an ability to add a 3D Markup to a snapshot on an SSVc comment. And then, they can reconstruct the original 3D View scene from the message.

The 3D View reconstruction requires information about the View state: camera zoom and rotation, hidden and shown 3D components, and so on. This information is contained with the graphic in a message. For DPN, restoring a 3D View scene necessitates the re-execution of the query used to define the View, parameters used, View mode, and camera location.

## View Modes

View Modes are a mechanism to visualize 3D geometry data using alternate colors and opacities. View Modes are enabled exclusively using the Query Processor API with the addition of Rendering Configurations; see sections Alternate Query Processing and Rendering Configurations respectively.



## Saved Views

Saved Views are a mechanism to restore an original 3D View scene from a modified 3D View scene by re-applying the functions that resulted in the original scene at the time it was saved. These include the selected Dynamic View Definition, used Parameter Values, selected View Mode, added parts and assemblies, and current camera position.

A Saved View includes only input information as opposed to the IDs, for example, of all displayed view files. As a result, a restored 3D View scene is not guaranteed to match an original 3D View scene because the results returned by QD execution may include a different result set.

Not all 3D View scene parameters are restored. For example, the Display Style, Exploded Increment, Measurements, and Cutting Planes are not included.

Given Saved Views will remain associated with all versions of a given target Item.

Saved Views are stored using the SavedView ItemType with source types derived from the Dynamic View Definition context ItemTypes collected under the SVTargetItem poly ItemType. On Dynamic View Definition creation, a context ItemType will be added to the Poly Sources list of the SVTargetItem ItemType, if it has not been already added. The SVTargetItem ItemType will be created during the DynamicModelConstrator AML package import.

### Important

If a Dynamic View Definition was created for an ItemType other than **CAD** prior to 12.0 SP8, the **Saved View** button may not appear in the **Dynamic Viewer** tab due to non-existing source type. In this case, the context item for this Dynamic View Definition must be added to the **Poly Sources** list of the **SVTargetItem** ItemType.

### Important

If a Query Definition or Query Definition Parameters are modified or removed after a Saved View is created, errors may be shown when restoring the View state.

The restoration of the camera position for the **Saved Views** can be turned on and off by the administrators using the `3D Visualization.RestoreCamera` variable. To restore the current



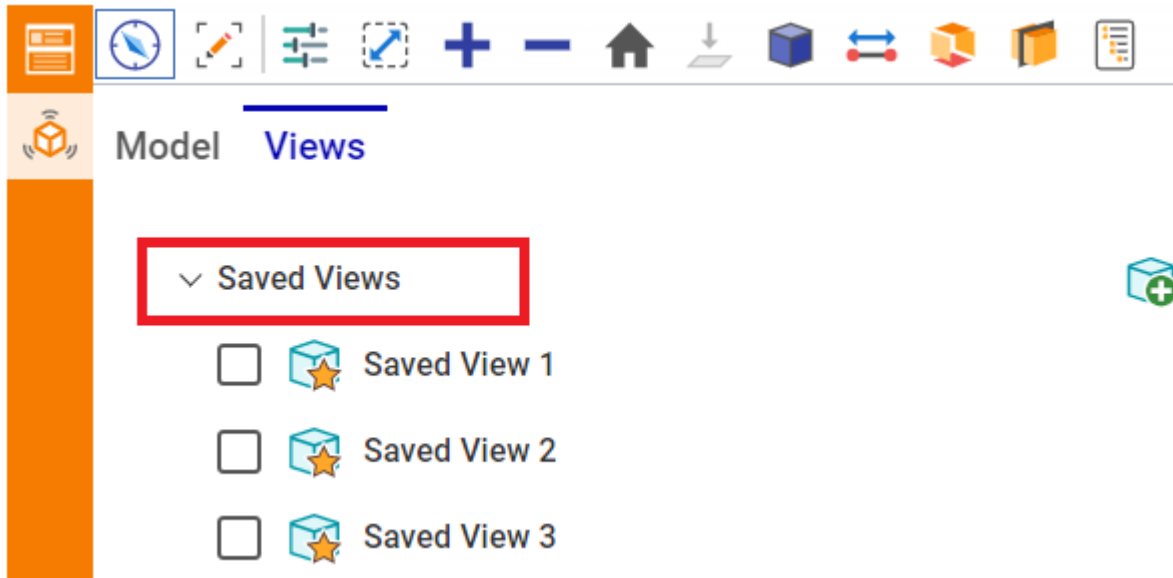
camera position of the Saved View in the 3D Viewer, set the variable to True. When the variable is set to False, the camera position of the Saved View in the 3D Viewer is not restored when the Saved View is rendered.

In future, end users will have the ability to configure sub section names under the Saved View section. This will enable users to organize the Saved Views.

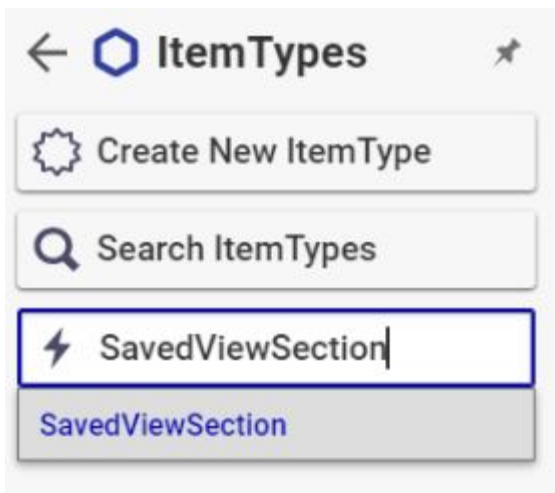


### Modifying Default Saved Views Label

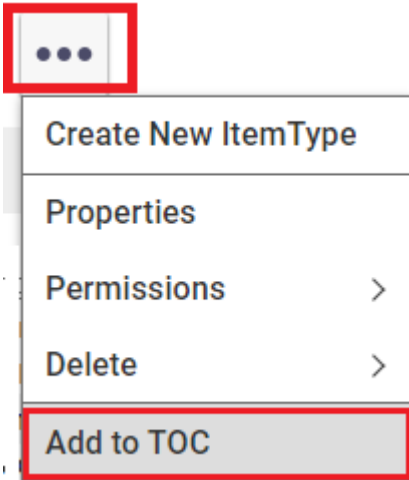
The following steps outline the process to change the name of the **Saved Views** label under the **View** tab of the 3D CAD model by root user:



1. From the Table of Contents, expand **Administration** and select **ItemTypes**.
2. In the **Quick Search**, search for **SavedViewSection** ItemType.







3. In the **SavedViewSection** ItemType, click **More** and select **Add to TOC**.



The **SavedViewSection** ItemType should appear in the TOC section.




# Contents

- > Administration
- > Change Management
- > Design
- ✓ Documents
  -  CAD Documents
  -  Documents
- > Extended Classification
- > My Innovator
- > Portfolio
- > Sourcing
-  Saved View Sections **

4. From the Table of Contents, click **Saved View Sections ItemType** and click **Search Saved View Section**. The **Saved View Section** grid view appears.



## Saved View Sections





 Search  Clear Simple  | Default     



	Section Order	Name	Default Section
			





5. Click **Search**.
6. Select **Saved Views** instance from the grid and click **Edit**.




**Saved View Sections**  


 **Search**  **Clear**   |  

	Section Order	Name	Default Section
			
	128	Saved Views	<input checked="" type="checkbox"/>

7. Change the name of the **Saved View** from the **Name** field.

**3D Saved Views**  

 **Save**  **Done**  **Discard**

 **Saved View Section**

**Name**

**Section Order**

**Default Section**

8. Click **Done** to save the changes.

To confirm the changes, go to the **View** tab of the CAD model.








Model Views

3D Saved Views



-  Saved View 1
-  Saved View 2
-  Saved View 3



## Digital Mockup

In Aras 3DV, a digital mockup is an ad-hoc arrangement of 3D component geometry on the 3D scene for analysis, review, or other purposes. End users can visualize collections of 3D assemblies, subassemblies, and parts in a manner that may be different from how these objects were defined within a CAD editor.

The Dynamic/Streaming Viewers provides the users with the abilities to:

- Add additional assemblies, subassemblies, and parts onto a single 3D scene
- Manipulate the position, orientation (by 3D rotation), and display of a selected 3D geometry.

Whenever an additional assembly, subassembly, or part is added to the Dynamic/Streaming Viewers:

- An associated Dynamic View Definition is re-executed using selected Parameters.
- The Tree Grid View and 3D View scene are refreshed.
- All defined View Modes and Parameters are applied to the context and added assemblies, subassemblies, and parts.
- No transformation is applied to the root of the added additional assembly, subassembly, or part.

An additional assembly, subassembly, or part is added to the TGV as a separate root item. The context item may not be at the top in the TGV because this grid is sorted using the sorting logic of an associated QD.



## Context Menu Functions

The Dynamic Viewer and Streaming Viewer supports context menu functions in the TGV and 3D View that can be extended as outlined in sections DynamicView Client-Side API and Customization Example with DynamicView API.



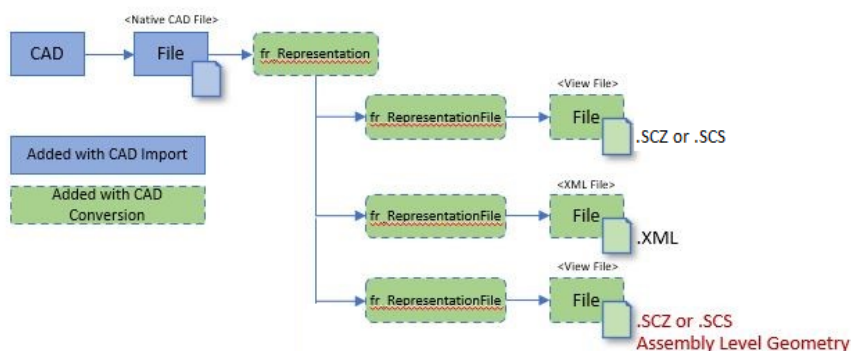
## Assembly Level Geometry Support for NX Files

The Dynamic and Streaming Viewers support the ability to view **Assembly Level Geometry** for **NX** files. The **NX Assembly** view files directly contain 3D geometry at the root level (assembly level), rather than parts that are normally used to define the assembly.

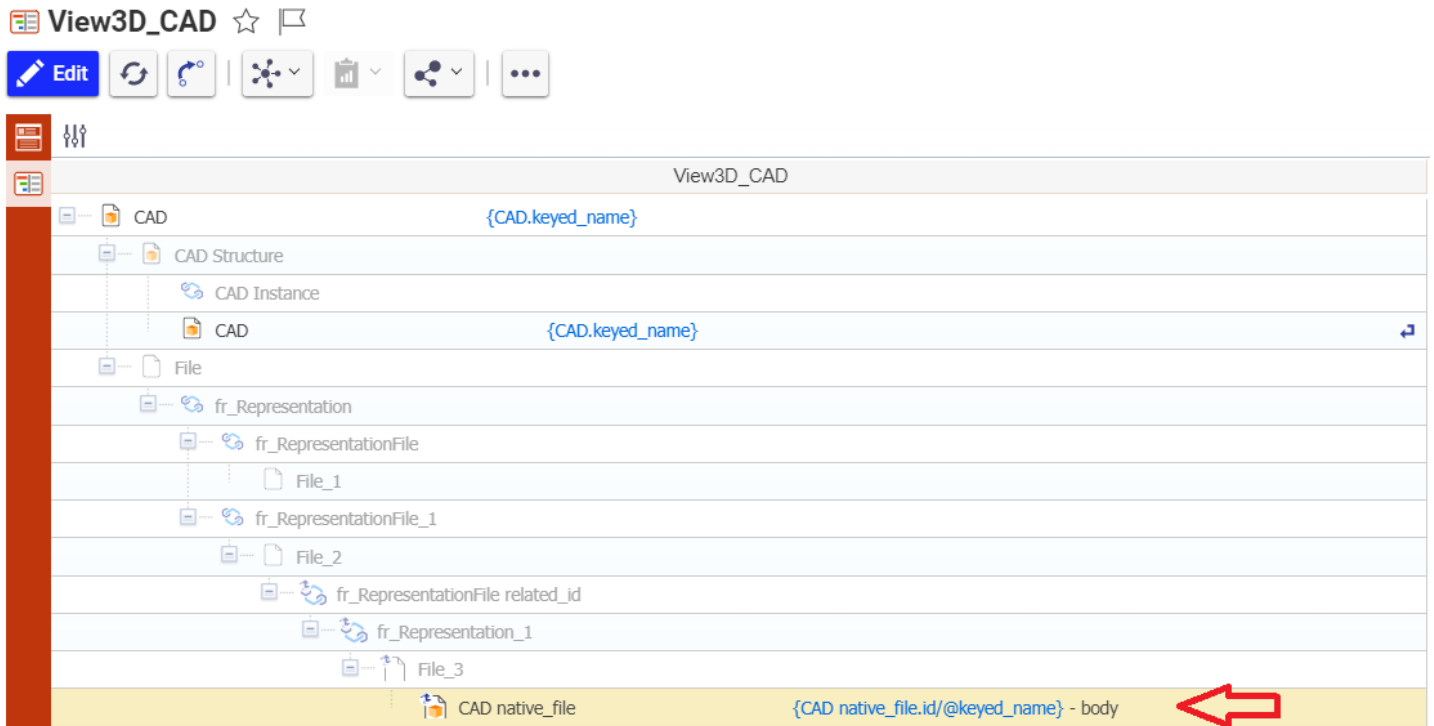
The native CAD assembly files can contain 3D component geometry. To create Assembly Level Geometry **FileRepresentation** view files for **NX** files, the **ConversionServerConfig** file in the root Aras Innovator folder is updated with the command-line option. The **FileRepresentations** view file is generated during the conversion process, and the default Query Definition supports viewing this structure in the Dynamic Viewer or Streaming Viewer.

The following figure describes the process.

## File Representations Assembly Level Geometry



The default Tree Grid View Definition for Dynamic and Streaming Viewers is modified with a row for **Assembly Level Geometry** view file. The new row contains a text called **body**.



The following steps outline the configuration to view the **Assembly Level Geometry** for **NX** files:

1. Install Aras 3D Visualization package. Refer to the Aras 3D Visualization 35 – Installation Guide.
2. In the root Aras Innovator folder, change the **Assembly Command** tag in the **ConversionServerConfig.xml** file to the following:

- **For Dynamic 3D Viewer:**

```
<AssemblyCommand dynamicEnabled="True" arguments="--
sc compute bounding boxes 'All' --input pdf template file
'C:\Aras\14sp10\Templates\Blank Template L.pdf' --output pdf
'%filepath%\%filename%.pdf' --output png '%filepath%\%filename%.png' -
--output png resolution '150x150' --output scs
'%filepath%\%filename%.scs' --output xml assemblytree
'%filepath%\%filename%.xml' --output prc '%filepath%\%filename%.prc' -
--background color '1.0, 1.0, 1.0' --output logfile
'%filepath%\%filename%' isolatedModelArguments="--output scs
'%filepath%\%filename%.scs' --import hidden object 'True' --import_pmi
false --output_logfile '%filepath%\%filename%_isolate.log' " />
```

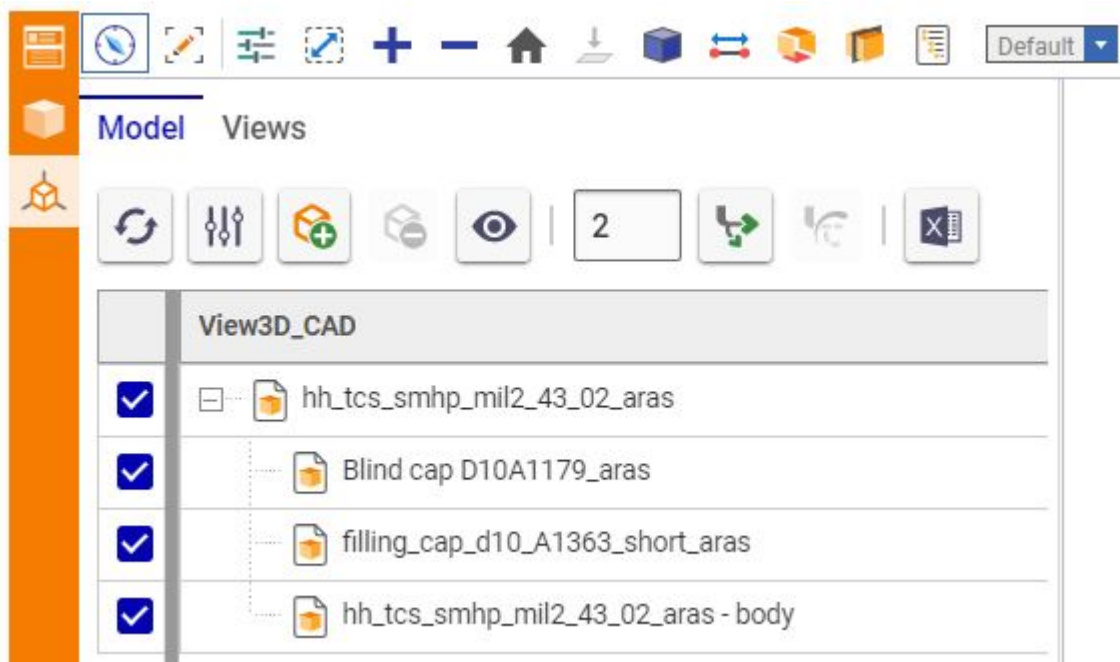


- For Streaming 3D Viewer:

```
<AssemblyCommand arguments="--sc_compute_bounding_boxes 'All' --
input pdf template file
'C:\Aras\14sp10\Templates\Blank Template L.pdf' --output pdf
'%filepath%\%filename%.pdf' --output png '%filepath%\%filename%.png' -
-output png resolution '150x150' --output xml assemblytree
'%filepath%\%filename%.xml' --output prc '%filepath%\%filename%.prc' -
-background color '1.0, 1.0, 1.0' --output logfile
'%filepath%\%filename%' " isolatedModelArguments="--sc create scz
'true' --output directory '%filepath%' --import hidden object 'True' -
-output sc '%filepath%\%filename%' --import pmi false --output logfile
'%filepath%\%filename%_isolate.log' " streamingEnabled="True" />
```

The Assembly **Level Geometry FileRepresentation** view files are created. After the above configuration step, if a user converts **NX** assemblies which contains a 3D geometry at the assembly level, an SCS or SCZ file is generated which represents the newly added geometry.

In the Default Dynamic View Definition, this file is represented as an assembly with associated child components and an assembly level geometry associated with a text called **Body**.





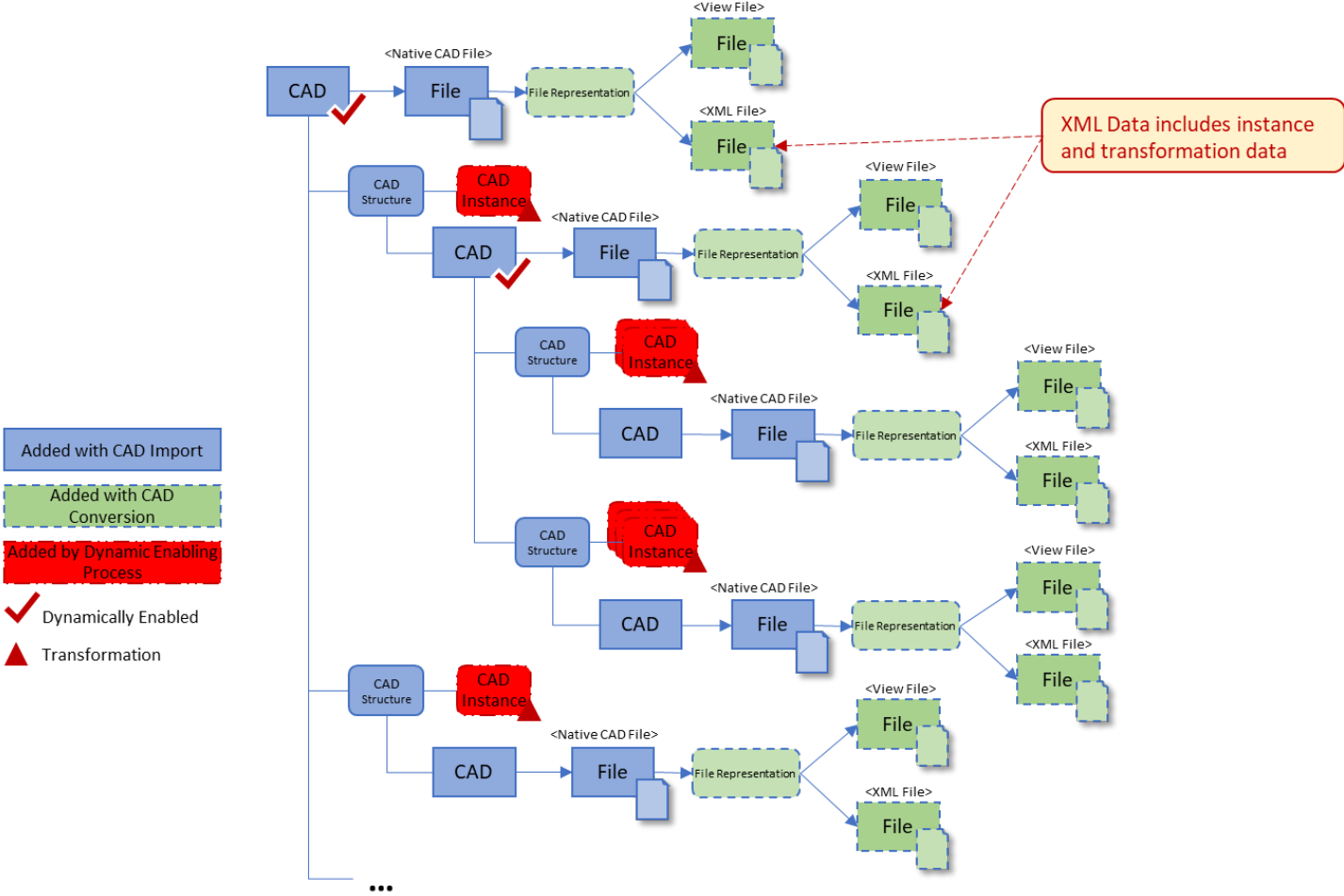
## Dynamic Enabling

There is an Action that, when used on a **CAD** Item assembly that is currently not Dynamically Enabled, will process the complete Assembly hierarchy to generate the necessary data to enable the assembly for viewing using the Dynamic Viewer. As noted in the following figure for the Dynamic Viewer to be used, **CAD Instance** data needs to exist in the CAD Structure that identify instances and their transformations. In addition, the **Dynamic Enabled** flag is set on each CAD assembly. The combination of this information will enable the Dynamic Viewer. The **Dynamically Enable Legacy CAD Items** Action is added so that any **CAD Structure** Item that was generated in Aras Innovator versions from 11.0 SP11 and higher and upgraded to Aras Innovator 12.0 SP4 and higher could be processed to add the **CAD Instance** Items so that those Assemblies could be visualized using the Dynamic Viewer.

### Important

Aras Innovator versions prior to 11.0 SP11 do not have the necessary data that the Dynamic Enable Action requires. For these environments, users will have to re-import CAD data to use the Dynamic Viewer.





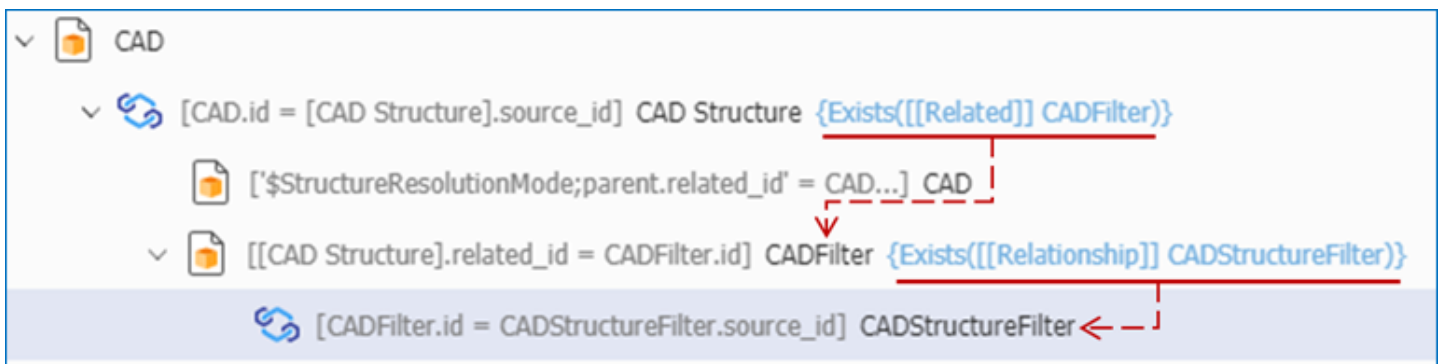
## Dynamic Enable Process

The Dynamic Enable process uses the XML data created from the Conversion Process (see Section CAD Data Model) to identify each Component Part and Sub-Assembly instance and its transformation matrix to generate CAD Instance Items for each level in the CAD / CAD Structure hierarchy. This XML data exists for all CAD data imported into Aras Innovator versions 11.0 SP11 and higher. The Dynamic Viewer uses the same view file data (SCS) to render the 3D geometry. Because the necessary information exists, the Dynamic Enabling process executes quickly as compared to the import and conversion process for new CAD data.

The Dynamic Enable process uses the Conversion Server to process the selected CAD Assembly asynchronously on the Server. As a result, the only status provided to the user is the Conversion Task Item (**Administration** → **File Handling** → **Conversion Tasks** in the TOC) that is subsequently created when the Action is executed. When complete, the selected Assembly and all sub-Assemblies will be dynamically enabled.

**The CAD ItemTypes are manually versioned. However, this process will not update the generation or revision of CAD Items.** Dynamic Enable Query

When processing Assemblies for *Dynamic Enabling*, a Query Definition is used to identify all sub-assemblies and sub-sub-assemblies and so on. The default **View3D\_DynamicallyEnablingCAD** Query Definition is used for this purpose. The query uses the 'Exists' Aggregate Function in the Where Condition applied to the CAD Structure, along with added Query Items that will identify the existence of related CAD Structures and CAD Items. The result will only process CAD Items in the CAD Structure hierarchy that are Assemblies – it ignores Component CAD Items. This is because the Dynamic Viewer only applies to Assemblies.



**Important**

Federated Properties cannot be used in Where clauses for a Query Definition.



## Enabling Legacy CAD Items to Work with Dynamic Visualization

The following steps outline the process to enable legacy CAD Items to work with dynamic visualization:

1. Enable the DynamicEnabler User. The DynamicEnabler User is required to update legacy CAD items only – it is not required to dynamically enable newly converted CAD items.
  1. Log into Aras Innovator as an administrator.
  2. Find and edit the dynamic\_enabler User item.
3. Set **Logon Enabled** to true and choose an appropriate password.

The screenshot shows the user management interface for the 'dynamic\_enabler' user. The 'Logon Enabled' checkbox is checked and circled in red. The 'Login Name' is 'dynamic\_enabler' and the 'First Name' is 'DynamicEnabler'. Other fields include Password, Confirm Password, Email, Fax, and Cell.

2. On the Aras Innovator Server, open the InnovatorServerConfig.xml file and add the following Operating Parameter:

```
<operating parameter key="dynamic_enabler" value="dynamic_enabler |
<Password>" />
```

3. In Aras Innovator, add the **DynamicEnabler User** Identity as a member of the Aras PLM Identity.

**Important**

This is done because DynamicEnabler needs Update access to all CAD items that need to work with Dynamic Visualization.

4. Select CAD Items from the database and run the **Dynamically Enable Legacy CAD Items** Action as needed.

## Creating Query and Tree Grid View Definitions

Dynamic visualization renders a 3D View based on the product of a query. The data model on which the query is executed needs to include the elements required for a 3D view:

- 3D Component geometry
- Instances
- The transformations necessary to place and orient each instance

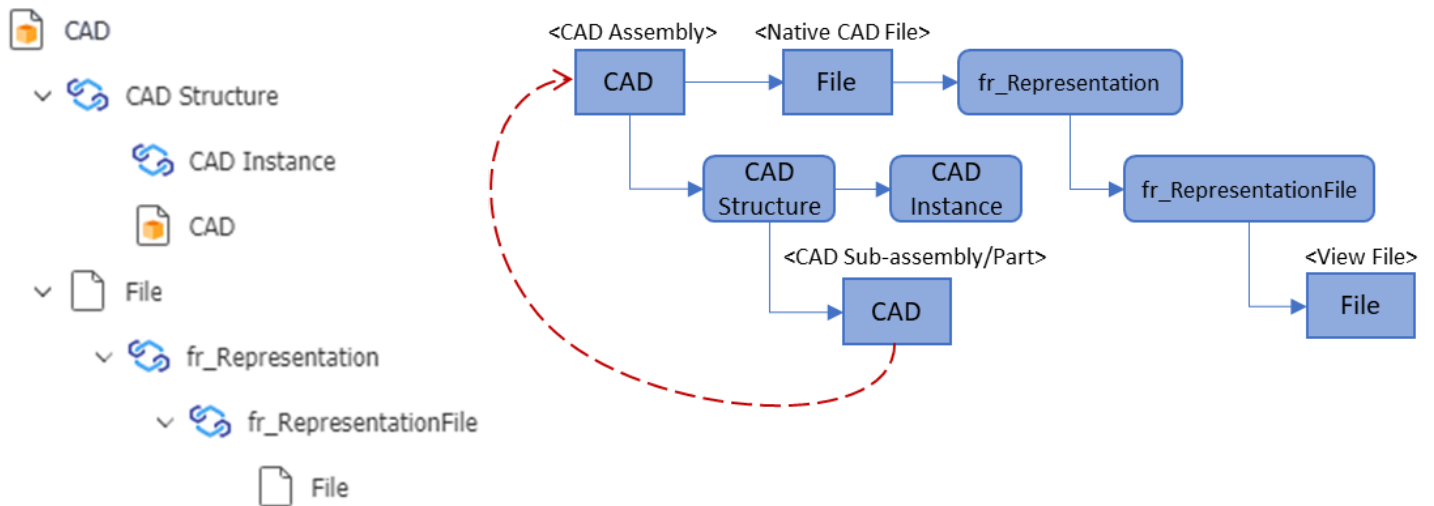
This information is contained in the out-of-the-box data model for CAD, CAD Structure, and CAD Instance ItemTypes. This section describes the default Query Definition, how to copy and customize it, how to use it in a Tree Grid View Definition, and how to 'register' this Tree Grid View Definition for use by the Dynamic Viewer.

## Query Definitions

Query Definitions are used by the Dynamic Viewer and Streaming Viewers to determine what should be included in a 3D View when the View is processed. There are restrictions on the Query Items included and the specific relationships for the Dynamic Viewer and Streaming Viewers to work properly. This section identifies the information that is required to render a dynamic view, where this information is located by default, the Query Definition that properly identifies this data, and areas that can be customized.



### CAD / CAD Structure Data Model



## Query Definition

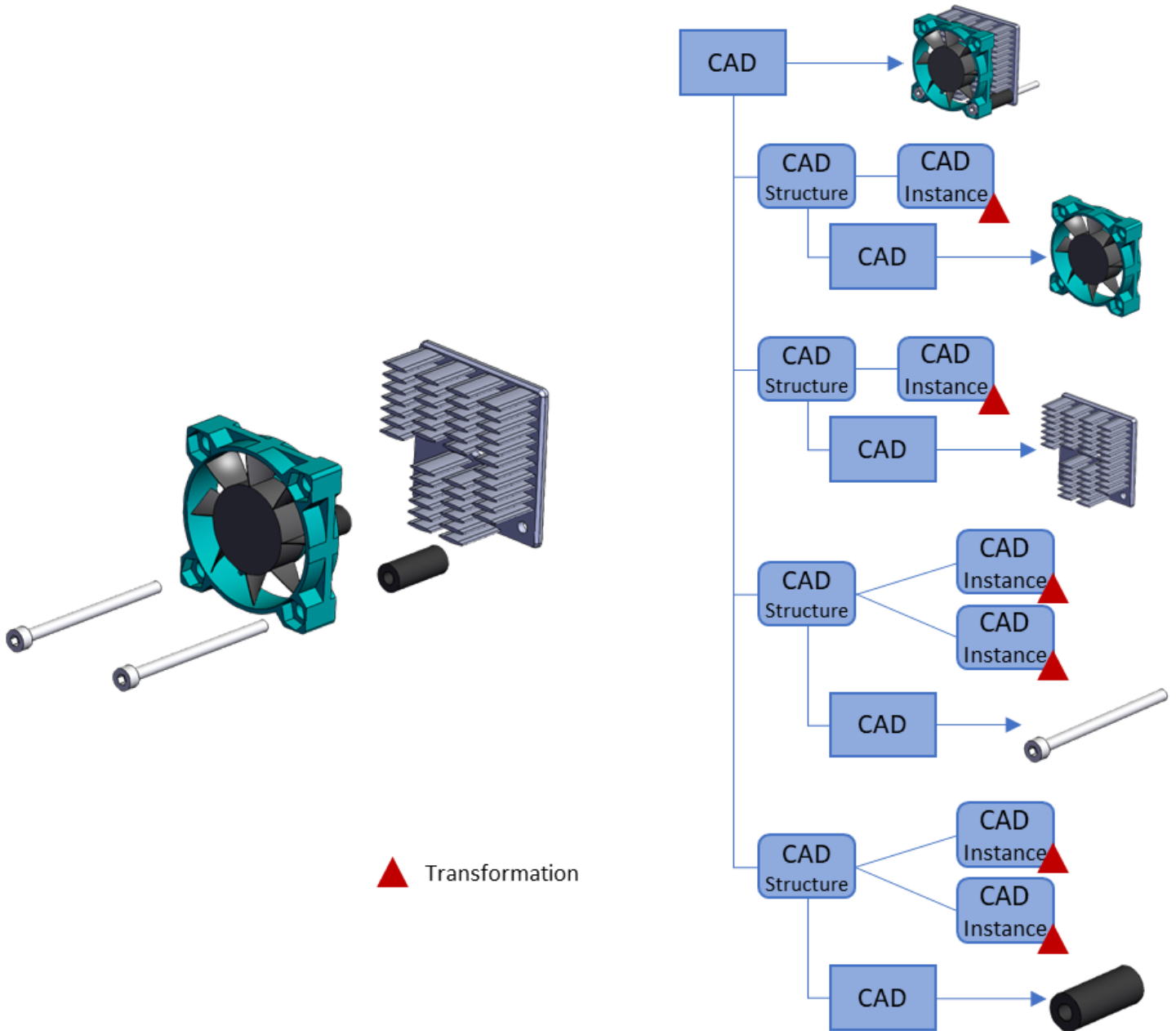
## CAD Structure Data Model

To render a 3D View, the following data is required:

- View files, for 3D component geometry
- Transformations, to determine where to place the 3D component geometry in 3D space
- Instances, to determine the number of each of the 3D components
- Assembly hierarchy, to determine the lineage of transformations to apply.

CAD Item in the CAD Structure. It is accessed as related content on the File Item used for storing the native CAD File. This 'related content' is referred to as a 'File Representation' because the generated View File is based on the associated native CAD file. Note also that the XML data generated from the conversion process (see Section CAD Data Model) is also stored as a File Representation. The Transformation is stored as a String but parsed as a 4X4 matrix of floating-point values<sup>1</sup>. This String Property is contained in the CAD Instance Relationship Item, which is directly associated with the CAD Structure Relationship. The number of CAD Instance Items determine the number of Instances of the related/child CAD Item within the parent/source CAD Assembly. The following diagram further illustrates the data model. Note that the File Representations are removed for clarity.





CAD Structure queries are processed top-down in a recursive manner. That is, CAD Items 'higher' in the CAD Structure hierarchy are processed before their children and so on. The order of processing determines the order of applying the transformations.

1. The interpretation of the matrix values is determined by the 'CAD Transformation Matrix Format' Variable (see Variables in the Administration folder in the main Table of Contents).



## Base Query Definition

This shows the Query Items for the Base Query Definition. These elements correspond to the diagram. The Query Definition named 'View3D\_CAD', accessible in the Table of Contents folder hierarchy **Administration** → **Configuration** → **Query Definitions**, is configured by default for the Dynamic Viewer and Streaming Viewers (see section [Default Tree Grid View Definition](#) for the default Tree Grid View Definition). This Query Definition can be used as a guide when creating alternative Queries for use with the Dynamic Viewer and Streaming Viewers.

Name	Context Item Type
View3D_CAD	CAD
<b>Description</b> Default Query Definition for Dynamic 3D viewer - DO NOT REMOVE	

### Important

The Base Query Definition should not be modified. It has default Permissions designed to prevent inadvertent removal or change. See section Customizing the Query Definition for a description of how to customize the Query Definition used for the Dynamic Viewer and Streaming Viewer.

The base query identifies what data is necessary for the required information described in section CAD / CAD Structure Data Model. It is important to understand that this Query Definition, when executed, will return the 'Latest' in Structure Resolution – see Section Structure Resolution) CAD Structure.

### Important

The CAD Structure Relationship is Hard Fixed by default



The properties given in the following table are required to be included in a Query Definition for the Dynamic Viewer and Streaming Viewer.

ItemType	Alias	Property
CAD <sup>1</sup>	CAD	id, keyed_name, x_min, x_max, y_min, y_max, z_min, z_max, config_id, is_current
CAD ConversionInfo		id, x_min, x_max, y_min, y_max, z_min, z_max
CAD Instance	CAD Instance	id, transformation_matrix
File	File_1	id, filename

The CAD Query Item, defined as the child of the root (context Query Item), ‘reuses’ the relationships and Properties of the root Query Item. This also defines a recursive Query whereby all Items of the child will be re-queried with the same Properties and related content as the root, and so on. By default, the keyed\_name Property value is used for the text in the Tree Grid View. The xyz\_min/max Properties are used to define the bounding volume for the associated geometry in the view file. It is used by the 3D View to help optimize the display priority of 3D component geometry when rendering and focus the camera when the view data is refreshed.

CAD ConversionInfo is a Null Relationship added to the CAD ItemType specifically to store information/metadata collected from the conversion process. Legacy environments stored the same information directly on the CAD ItemType. Thus, processing the results from an execution of a Query Definition must first refer to the CAD ConversionInfo relationship and then the CAD Item if this relationship does not exist.

The CAD ConversionInfo Null Relationship stores metadata processed during the CAD conversion process, including bounding box content.

**Important**

Legacy environments used the CAD ItemType to store data from the CAD Conversion process. The **CAD ConversionInfo** Relationship was added to obviate the need to update a CAD Item (and thus lock it) during the conversion process.

The CAD Instance Query Item should include the id and transformation\_matrix Properties. As noted, the transformation matrix is used to position the view file geometry of the related child CAD Item.



The File Query Item that refers to the view file (note the Alias name - *File\_1* - in the table) should include the id and the filename Properties. These values are used to form the proper URL to the vault so the 3D Viewer can retrieve the view files during processing.



## Customizing the Query Definition

The Base Query Definition should not be modified. Doing so risks disabling Dynamic Visualization entirely. Instead, the best approach to creating alternative/custom queries is to copy this Item and modify the copy to adjust/add whatever changes are necessary. This section identifies the scope of changes that are permissible and how to make those changes.<sup>1</sup>

### Adding Properties

Any Property can be added to the base Query Definition if the Core Properties (see Section [Base Query Definition](#) ) are not removed. In addition, there are no restrictions on the set of Properties added to additional related Query Items. Properties are mapped to Tree nodes or columns in the Tree Grid View Definition (see Section [Tree Grid View Definitions](#) ) Doing so exposes the values of these Properties in the Dynamic Viewer or Streaming Viewer.

#### Important

Properties must be added to the Query Definition for them to be mapped in a Tree Grid View Definition.

### Adding related content

Related ItemTypes – that is, the ItemTypes that are related to any ItemType via an Item Property – can be included in a Query Definition for the Dynamic Viewer and Streaming Viewer. As noted in Section 3D Visualization Navigation Tool, this mechanism enables users to see content related to parts displayed in the 3D View. A related ItemType is added as a separate Query Item in the Query Definition.

#### Important

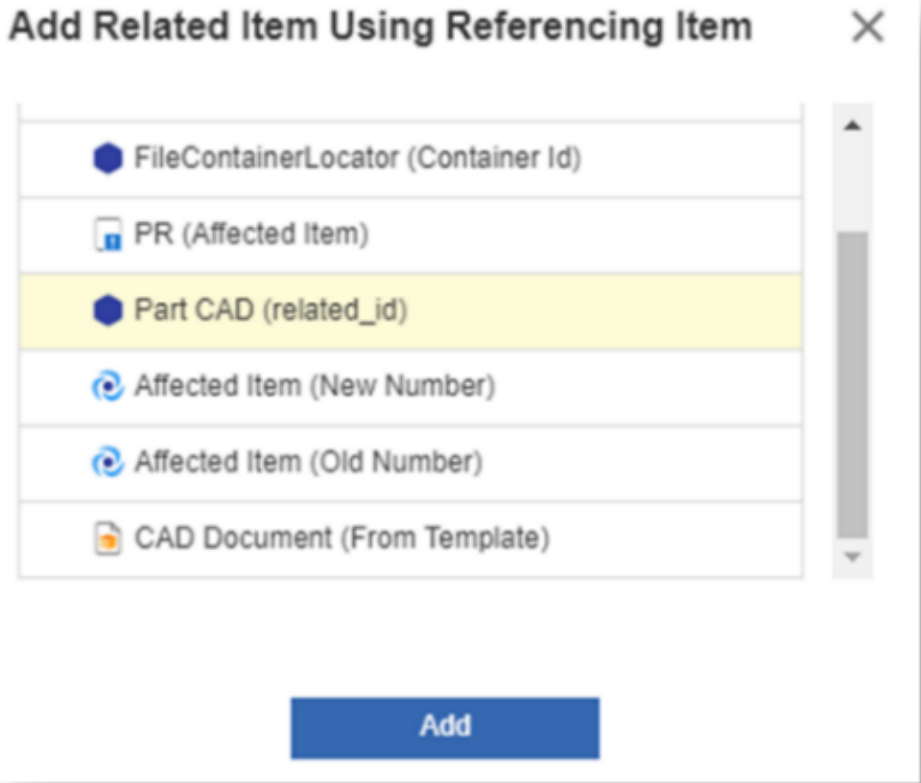
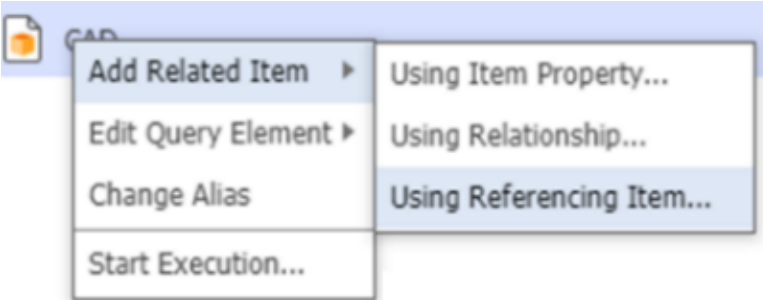
Query Items must be mapped in the Tree Grid View Definition for related Items to be included in the Tree Grid View.

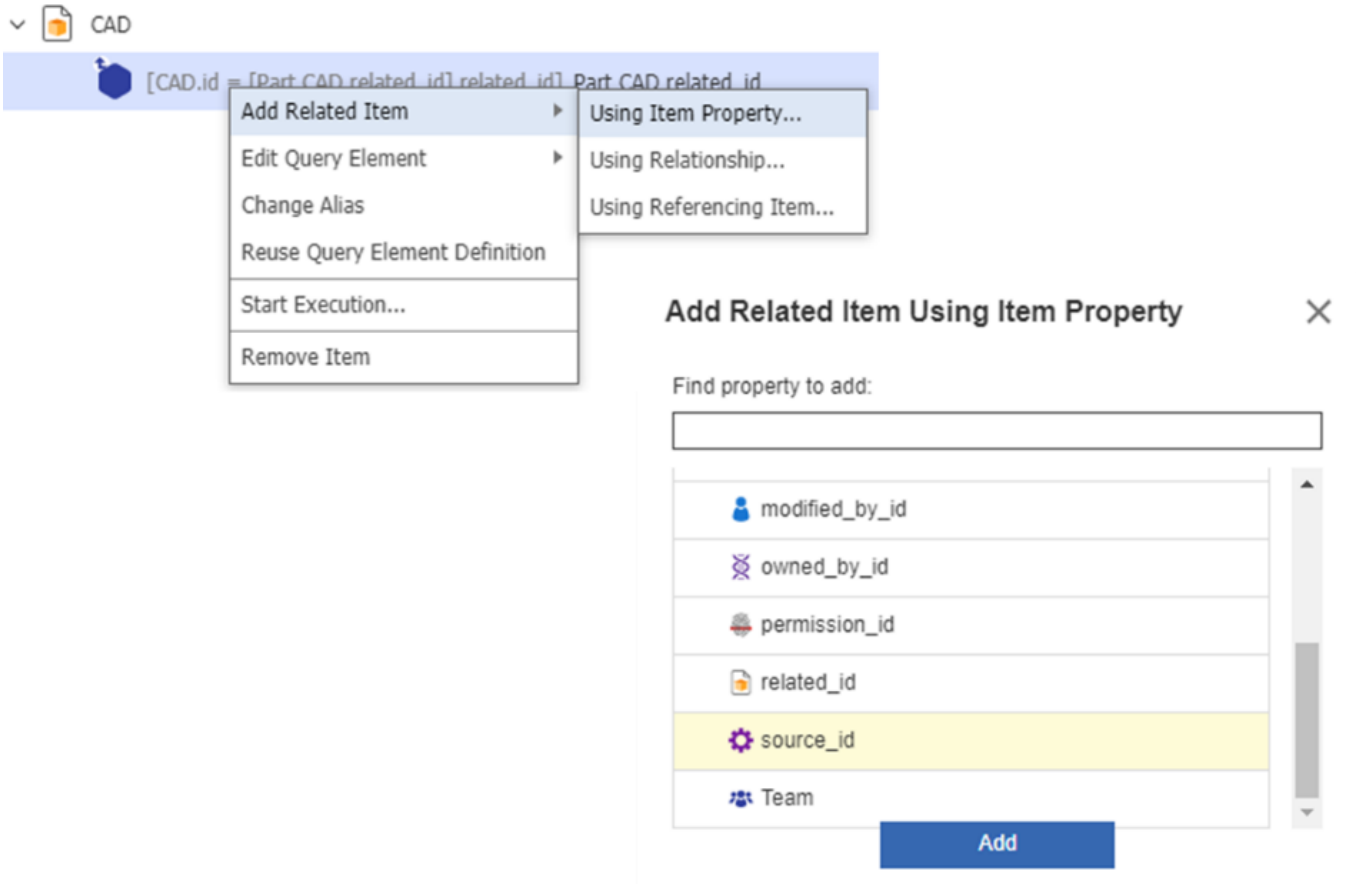
Any related ItemTypes can be included but it's important to note that the Base Query Definition must not be altered and the context ItemType must be associated with the CAD ItemType. Related ItemTypes use a Join Condition to determine how rows from the two associated ItemType tables



should relate. By default, these Join Conditions use the id Property of one of the Query Items in the Join. This Join Condition can be altered, but users should understand the ItemType data model they are referring to when doing so.

Since the Part ItemType represents a core Business Object in PLM, it is likely that users will want to see the Part Items associated with the CAD Items returned in the Base query. Parts are associated with CAD Items by default using the Part CAD relationship. Part Items, using this relationship, are the source and CAD Items are the related. Thus, if users want to include the Part CAD relationship as a 'referencing Item' in the Query Definition, they need to include the Part Item using the following steps as described by Figure 16 and Figure 17. Similar steps can be used when adding content that references the selected Query Item – that is, has an Item Property that 'points to' the ItemType of the selected Query Item.

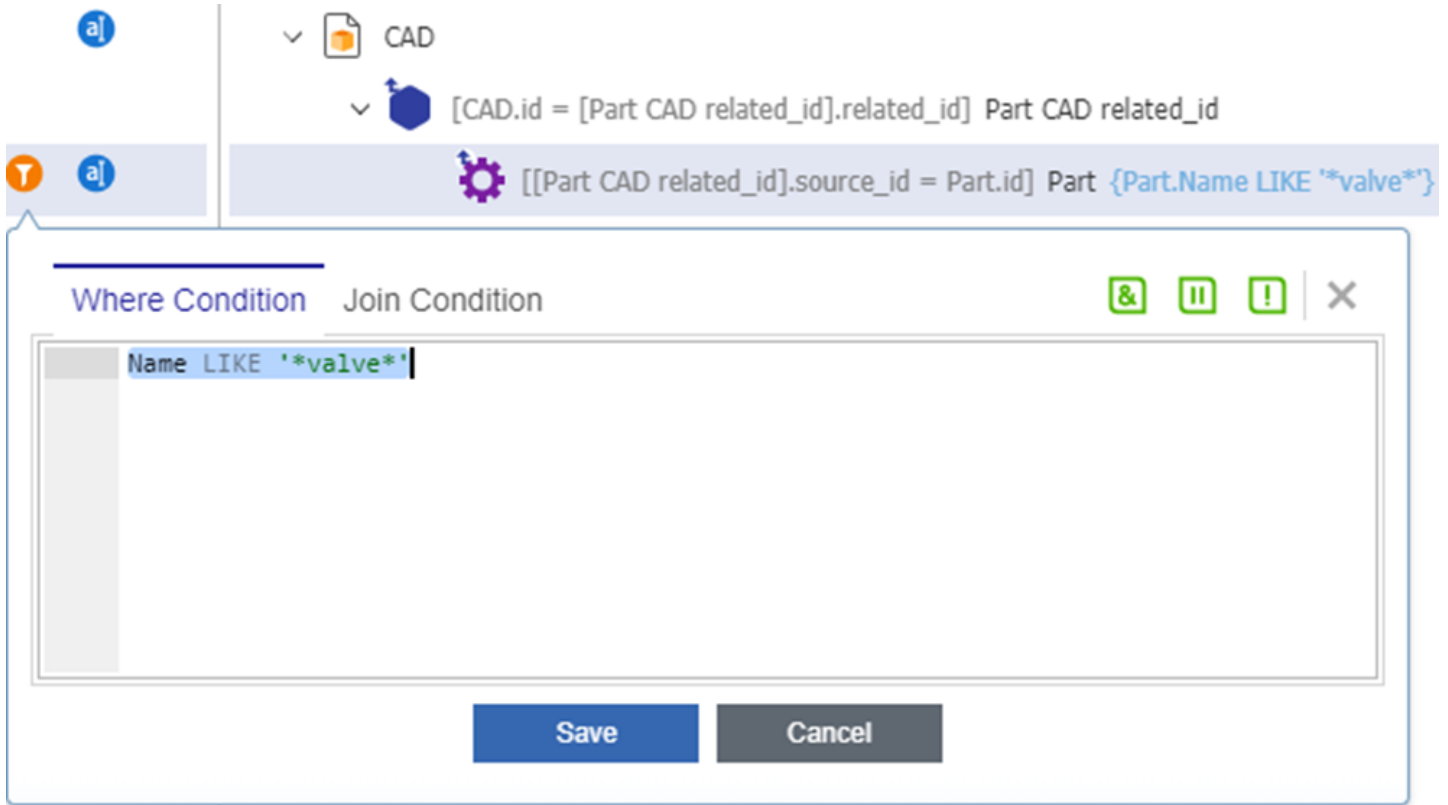




### Conditions and Filters

Conditional logic can be used to filter the content that is returned from the execution of a Query Definition. Where Conditions on the Query Items are used to add conditional logic. For example, the following Where Condition, applied to a related Part Item, results in only those Parts with the String 'valve' included somewhere in the Part Name being included in the query results.





**Important**

Properties used in Condition statements should be included with the Query Item. For example, the Name Property would need to be included with the Part Query Item for it to apply in this condition. Note also that the example in above figure will only filter Part Items, the Part CAD relationship Items will still be included in the query results.

**Important**

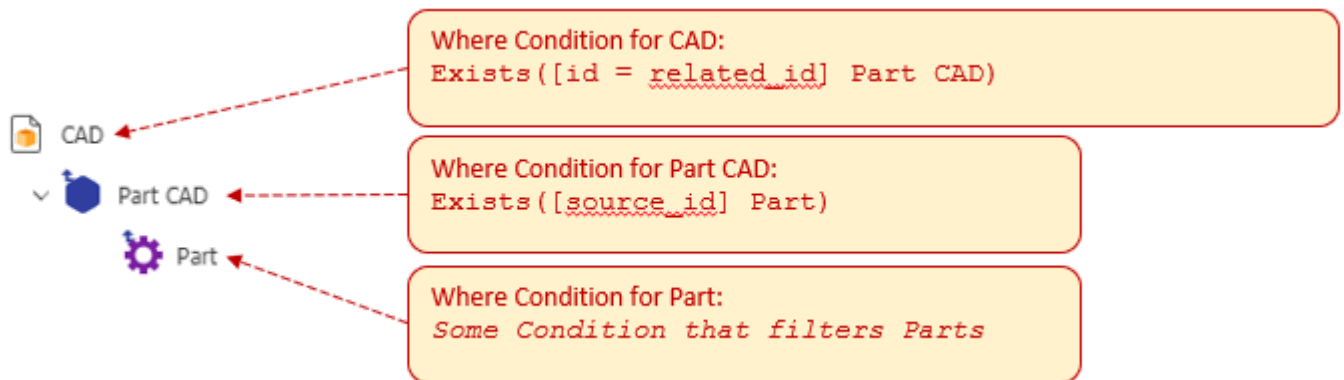
Federated Properties cannot be used in a Where Condition.

Conditional logic applied to any ItemType that is added to the Base Query Items can be done without affecting the Dynamic Viewer or Streaming Viewer. However, Administrators must use caution when applying conditions to the Query Items that are part of the Base Query Definition. For this purpose, the following information should be referenced:

1. Conditions that filter an assembly will result in all descendant CAD Items being filtered.



2. It is possible to filter CAD Items based on conditions applied to related content. For example, to filter component CAD Items based on conditions applied to related Parts:



- o The Condition applied to the Part Query Item can be whatever logic is necessary to isolate specific Part Items.
- o The Condition applied to the Part CAD Query Item uses the 'Exists()' function. This function effectively applies a SQL Inner Join between the Part CAD and Part Tables. The result is the rows for the Part CAD Items are reduced to only those with related Part rows in this case. That is, only Part CAD Items are returned if there are related Part Items returned which are filtered by some logic.
- o The Condition applied to the CAD Query Item needs to only associate the related part filter to CAD Items that represent components (non-Assembly Items). To do this, include the condition on the existence of the Part CAD Items. If a row is included, then include the CAD Item as well.

**Important**

Federated Properties cannot be used in Where Conditions.

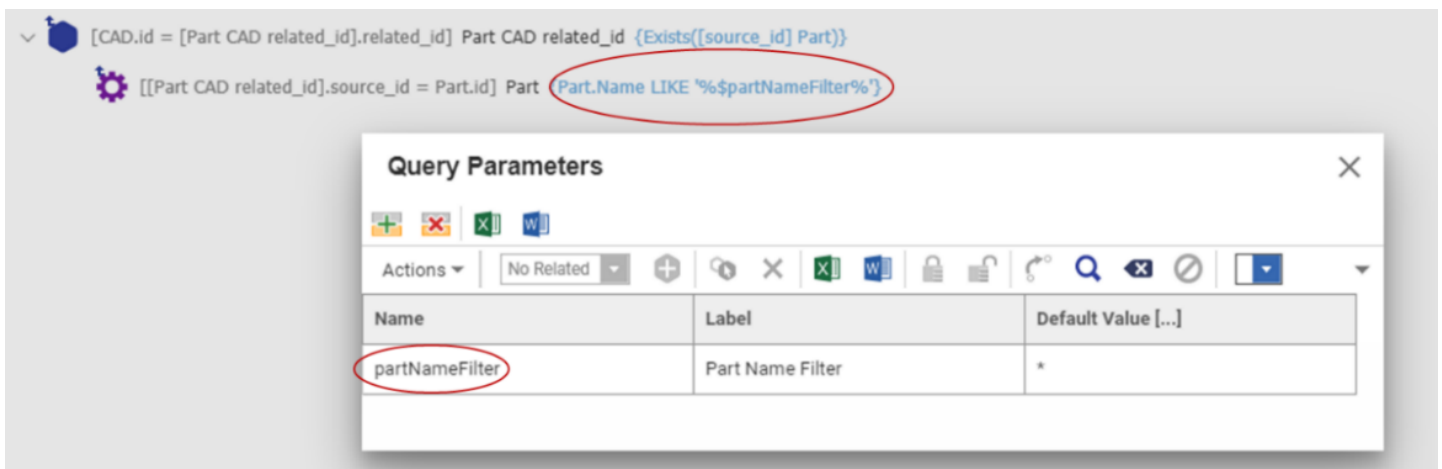
3. Use caution when applying filters to CAD Query Items. They are reused and exist to execute recursive sub-queries. In this case, the CAD Query Item represents both Assemblies and Components. Applied Conditional logic needs to consider that when Parts in a Dynamic View are placed, they require the application of the full lineage of Transformations from the root CAD Item to each leaf Component CAD Item.
4. If Conditions are applied to CAD Instance Query Items, instances of the related CAD Item are also filtered.
5. The Base Query Definition has a Condition that filters the File Item (view file) related by a File Representation based on the Representation 'kind'; which is SCS. If this Condition is modified, it may result in the corresponding 3D geometry being removed.

**Query Parameters**

Query Parameters used in Query Definitions provide the ability to add 'placeholders' for actual values used in Conditional statements. This is useful when there is a need to provide some level of control for additional filtering on the part of the end-user. It is possible to use a Parameter in place of the



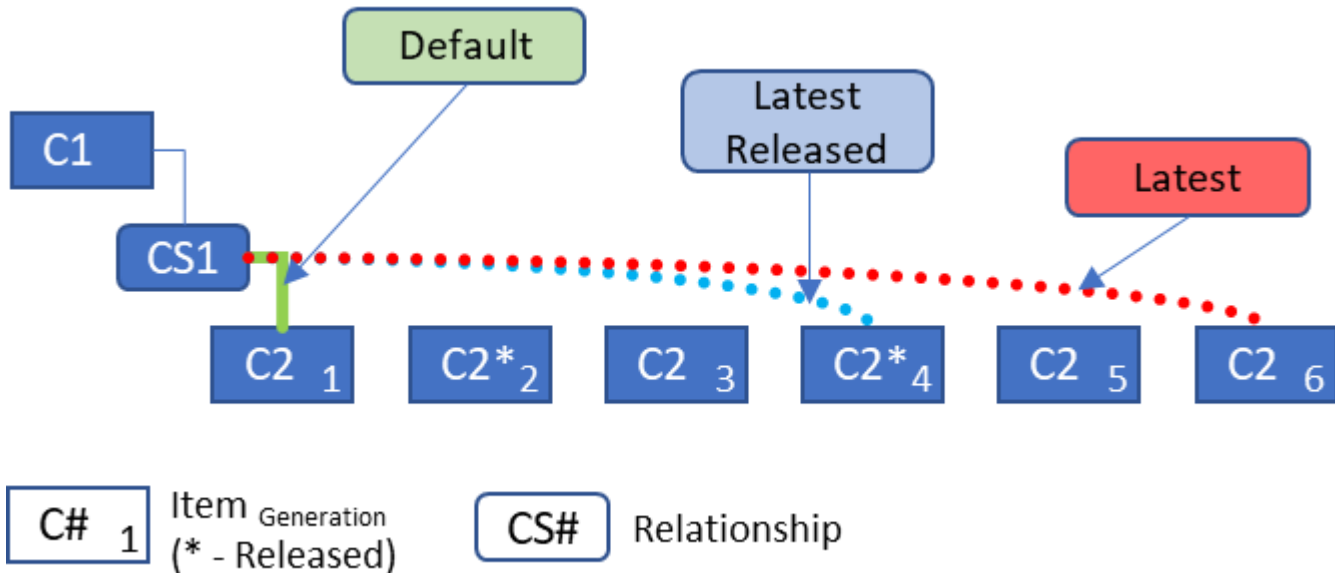
hard-coded value ‘\* valve \*’. In this case, Users can provide any value they would like to use in place of the default to filter based on the name of the related Part. The below figure shows the added Parameter ‘ partNameFilter ’ which can then be used in place of the value ‘ valve ’. This example uses the default Parameter value of ‘\*’ which is a wildcard character that will result in all names chosen in this case. Also, for this example the Parameter is placed in between two ‘%’ characters which will cause the name value given to be valid within the actual Part Name for the Part Items.<sup>7</sup> For example, a Parameter value of ‘ intake ’ would match Part Names: ‘Intake Valve’, ‘Left Intake Spring’, ‘Right Intake’ and ‘Intake’. Parameters are added using the Query Parameter Dialog and referenced within a Condition Statement by preceding the Parameter Name with a ‘\$’. Query Parameters must be added to the Tree Grid View Definition to be exposed to the user.



### Structure Resolution

Query Parameters enable another query mechanism to control the display of content – Structure Resolution. Structure Resolution is used to identify a specific generation of an Item. This is especially useful for CAD structures since the CAD Structure Relationship is fixed by default. This means that a CAD Structure hierarchy will be based on the CAD Items and specific relationships created from the CAD Connector (see Section CAD Data Model). Updates to individual CAD Items do not necessarily involve an update to the CAD Structure Relationship. As a result, Assembly CAD Items will ‘point to’ a generation of a CAD component that isn’t the latest.





Structure Resolution focuses on the Released state of generations of Items. It can be applied based on the following conditions/Modes:

- **Default:** Uses the generation of the Item that the Relationship Item includes. That is, it will be the Item with an ID matching the `related_id` Property of the Relationship.
- **Latest:** Uses the generation of the Item with the highest value.
- **LatestReleased:** Uses the generation of the Item with the highest value which is also Released.
- **Latest Released or Latest:** Uses the generation of the Item with the highest value which is also Released or the Latest if there are no Released generations.

### Enabling Structure Resolution

Structure Resolution is enabled for the Base Query Definition (Section Base Query Definition). However, to incorporate it a new Query Definition must be created – as defined earlier in this section – and Structure Resolution enabled for it. The following steps describe how to add Structure Resolution to a Query Definition that derives from the Base Query Definition.

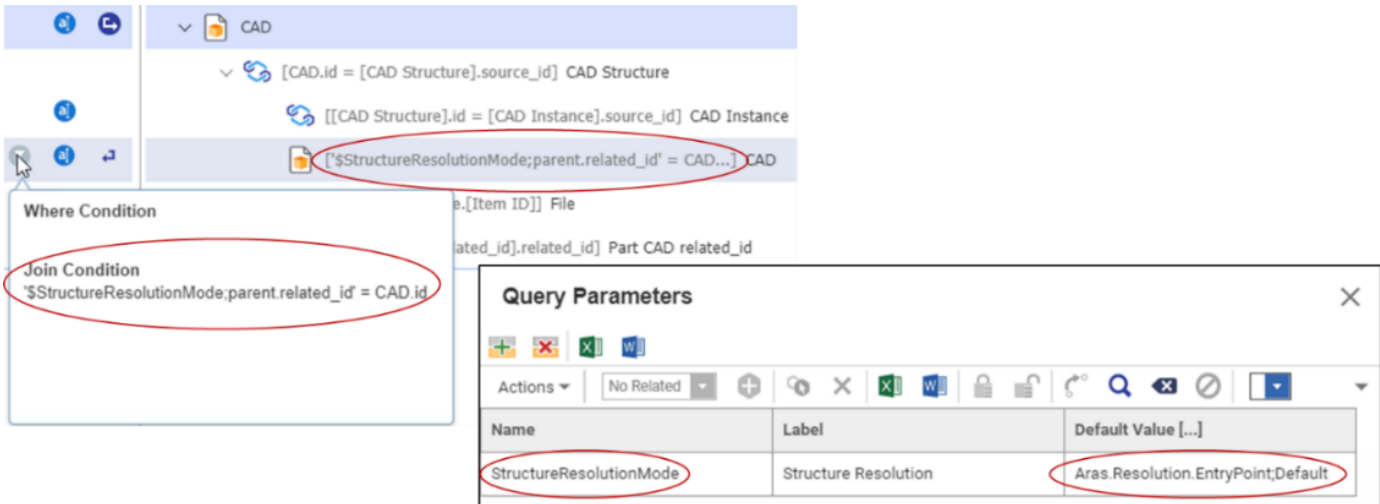
#### Step 1: Create Structure Resolution Mode Parameter

Applying a specific Structure Resolution Mode ('Default', 'Latest', etc.) is done using a Query Parameter. Using the Query Parameters Dialog, add a Parameter to be used for Structure Resolution Mode. For example `'StructureResolutionMode'` and assign the default value `'Aras.Resolution.EntryPoint;Default'`.



**Important**

Be sure to check the syntax of the Default Value for the Parameter. It is used to select a specific List Item as set in the Tree Grid View Definition and described in Step 4.



**Step 2: Incorporate the Query Parameter**

The added Query Parameter must be incorporated into the Query Definition in order for it to be used/exposed to the end-user. Add it as part of the Join Condition of the Child CAD Query Item as shown in the figure. Replace the default Join Condition – ‘ [CAD Structure].related\_id = CAD.id ’ with the following – ‘ '\$StructureResolutionMode;parent.related\_id' = CAD.id ’ .

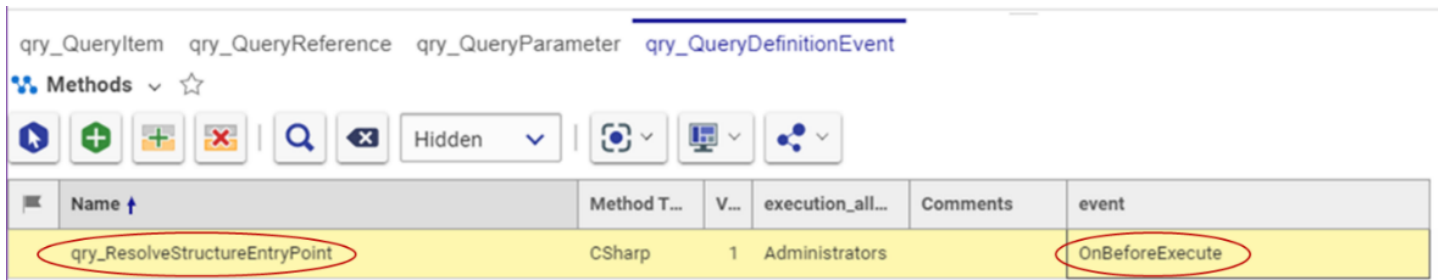
**Important**

Be sure to check the syntax of the Join Condition. It must be as specified above (without the outer quotes). This assumes the value StructureResolutionMode was used for the Query Parameter Name. The Structure Resolution Mode Query Parameter can only be inserted once in a Query Definition. It cannot be used/applied to more than one Query Item.

**Step 3: Add Resolve Query Entry Point Method**

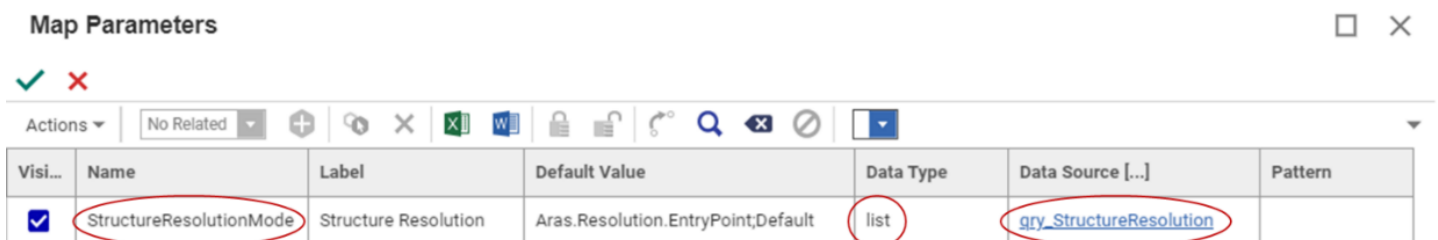


Show the Relationship Tabs for the Query Definition in the Form View if they are not shown. This can be done by selecting Tabs On for the Default Structure View in the `qry_QueryDefinition` ItemType form. Once shown, select the `qry_QueryDefinitionEvent` Relationship Tab. A Method needs to be added to the OnBeforeExecute event. To do this, select the Select Items Icon to create a Relationship and select the `qry_ResolveStructureEntryPoint` Method and assign it to the OnBeforeExecute event as shown in the figure.



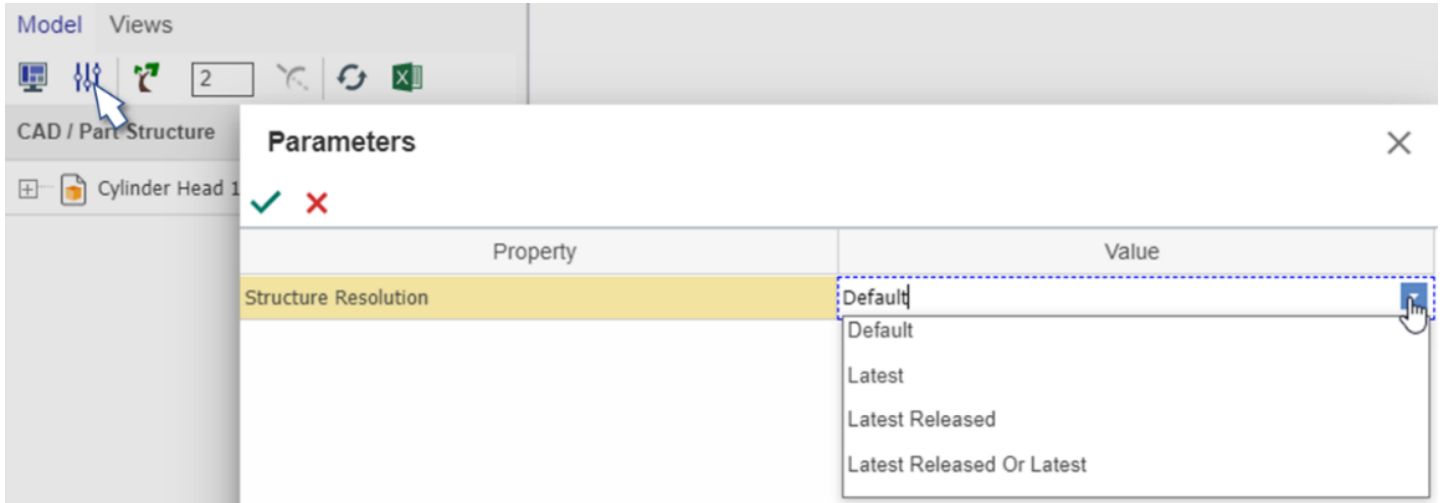
#### Step 4: Add the Query Parameter to the Tree Grid View Definition

For the Query Parameter to be used, it must be enabled (made visible) in the Tree Grid View Definition used for the 3D View. Open the Map Parameters Dialog in the Tree Grid View Definition within the Editor View. Select the **Visible** check box for the `StructureResolutionMode` row. The **Data Type** is a `List` and the **Data Source** is the List `qry_StructureResolution`.



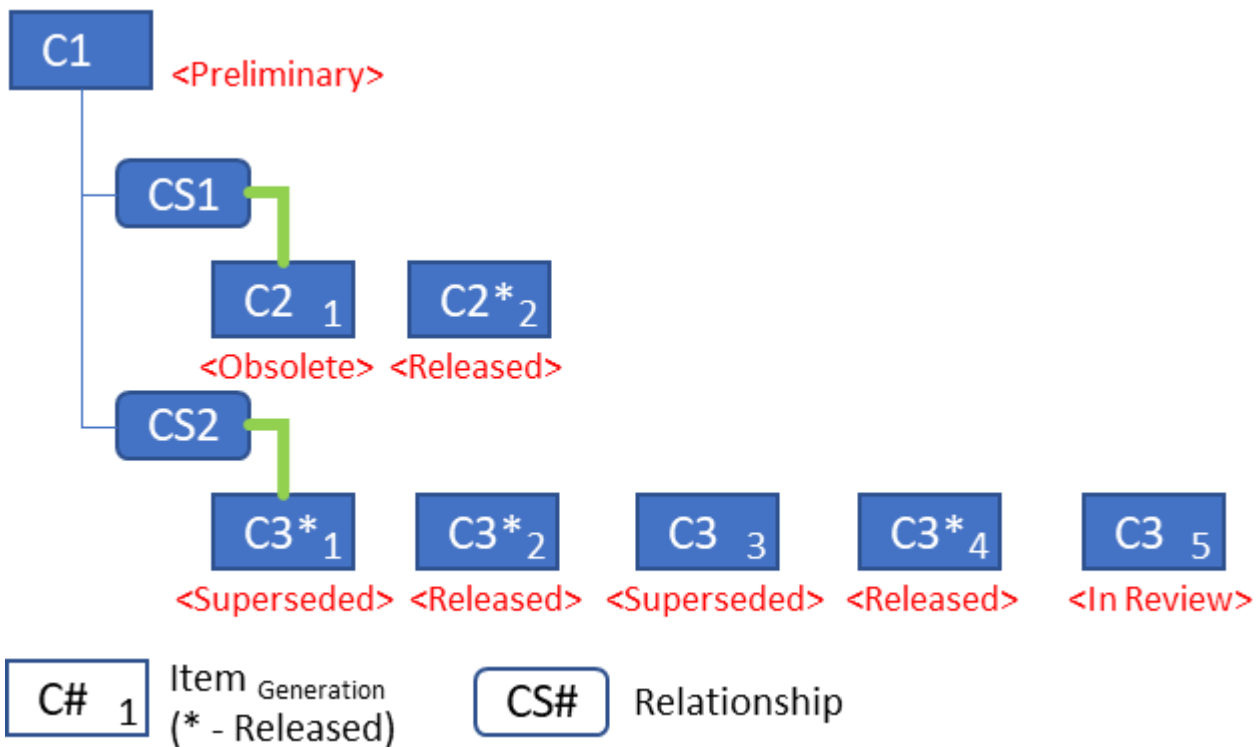
With this Configuration, users will be able to select specific Resolution Modes by opening the Query Parameters Dialog in the Tree Grid View used within the 3D Viewer.





### Applying a Where Condition Against Resolved Items

It is possible to add a Where Condition to a Query Item that is used for Structure Resolution (e.g., CAD). Doing so allows the Administrator to apply an additional condition (filter) when resolving the Structure of Items with fixed Relationships (such as CAD / CAD Structure). For example, assuming the CAD Structure in the following diagram:



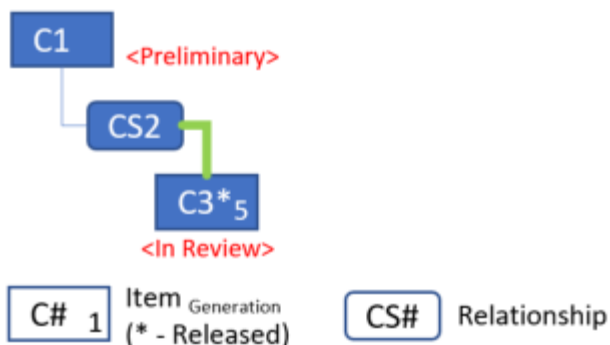
To use a Structure Resolution of 'Latest' and apply an additional condition to only include CAD Items that were In Review (that is, with a CAD.State = 'In Review') then Where and Join conditions like the following could be added to the Query Item:

The screenshot shows a query editor interface. At the top, there are several query items listed: 'CAD {CAD.State = 'In Review'}', '[CAD.id = [CAD Structure].source\_id] CAD Structure', and a highlighted item '[\$StructureResolutionMode;parent.related\_id;ApplyC...] CAD {CAD.State = 'In Review'}'. A dialog box is open, titled 'Where Condition' and 'Join Condition'. The 'Join Condition' tab is selected, and the text area contains the following condition: `'$StructureResolutionMode;parent.related_id;ApplyChildWhereMode=ResolutionTargetResolved' = CAD.id`. At the bottom of the dialog are 'Save' and 'Cancel' buttons.

Note the addition of the left-side of the Join Condition:

```
'$StructureResolutionMode;parent.related_id;ApplyChildWhereMode=ResolutionTargetResolved' = CAD.id]
```

Executing this Query Definition, with Structure Resolution of 'Latest Released' with return.



In this example, only the CAD Item – CS3, version 5 is returned because of the added Where Condition of ' CAD.State = 'In Review' .

## Tree Grid View Definitions

Tree Grid View Definitions are used to specify how information returned from the execution of a Query Definition will be displayed. Tree Grid Views are a combination of a Tree View – with the ability to display collapsible hierarchies of data in parent/child relationships – and tables – with columns of individual cells showing various static text and Property values. The Tree Grid View Definition configures the elements of the Tree Grid View; namely nodes for the Tree View (text and icon), columns with column labels, and mappings between Properties and column cells. Tree Grid View Definitions are associated with one, and only one Query Definition. This section describes the default Tree Grid View Definition, how to create custom Tree Grid View Definitions, how the Tree Grid View is used in the Dynamic and Streaming Viewers, and how to configure for automatic query execution and refresh when the Dynamic View is opened.



### Default Tree Grid View Definition

The default Tree Grid View Definition uses the Base Query Definition (see Section Base Query Definition) to display the hierarchy of CAD Items only. It is a similar configuration (in view only) to the CAD Item Display used by the Monolithic Viewer (see Section Monolithic vs Dynamic Viewer). For Aras Innovator, the Tree Grid View Definition named 'View3D\_CAD', accessible in the Table of Contents folder hierarchy **Administration** → **Configuration** → **Tree Grid Views**, is configured by default for the Dynamic and Streaming Viewers. This Tree Grid View Definition can be used as a guide when creating alternative Views for use with the Dynamic and Streaming Viewers.

<b>Name</b>	<b>Query Definition</b>
View3D_CAD	<a href="#">View3D_CAD</a>
<b>Context Item Type</b>	
<a href="#">CAD</a>	
<b>Description</b>	
Default Tree Grid View Definition for Dynamic 3D viewer - DO NOT REMOVE	
<b>Max Visible Children On Expand</b>	<b>Linked Toolbar/Context Menu</b>
100	<a href="#">View3D_CAD Presentation Configuration</a>
<b>Max Grow Levels</b>	
2	
<b>Auto Grow On Refresh</b>	
<input type="checkbox"/>	

#### Important

The Default Tree Grid View Definition should not be modified. It has default Permissions designed to prevent inadvertent removal or change. See section Customizing the Tree Grid View Definition for a description of how to customize the Tree Grid View Definition used for the Dynamic and Streaming Viewers.



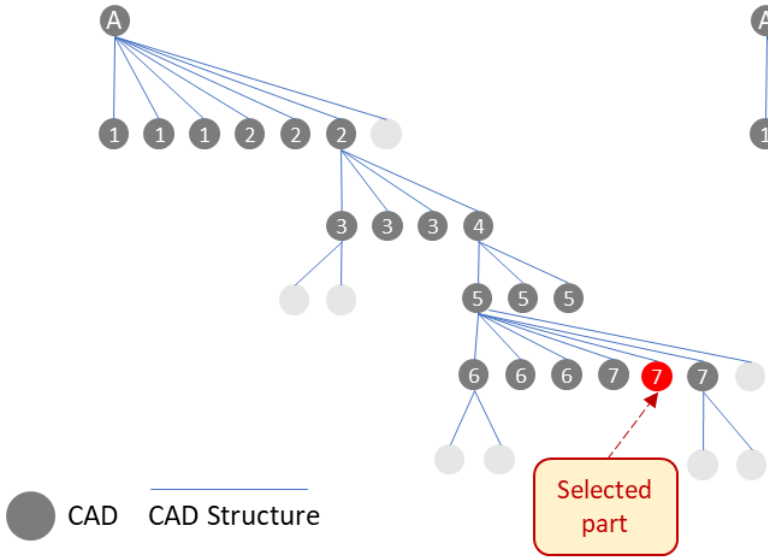
The **Max Visible Children On Expand** field is used to specify the maximum number of peer Items to query and display in the Tree Grid View when the view is refreshed or a related Item is expanded. Peer Items refer to the direct child Items for any parent Item. If there are more Peer Items that exist for any parent Item, the **show more** link is included in the Tree View portion of the Tree Grid View. Selecting this link displays the next set of peer Items and so on.

The value used for the **Max Visible Children On Expand** setting affects the performance of **3D View** → **Tree Grid View** selection synchronization. The reason has to do with the algorithm used to query down to the CAD Item that is associated with the selected 3D component geometry in the view. The larger the breadth of the assembly (that is, the larger the number of CAD Items at any given level in the CAD Structure hierarchy) the larger the potential for multiple simultaneous queries to be executed as the associated 'leaf' node in the Tree View is uncovered.

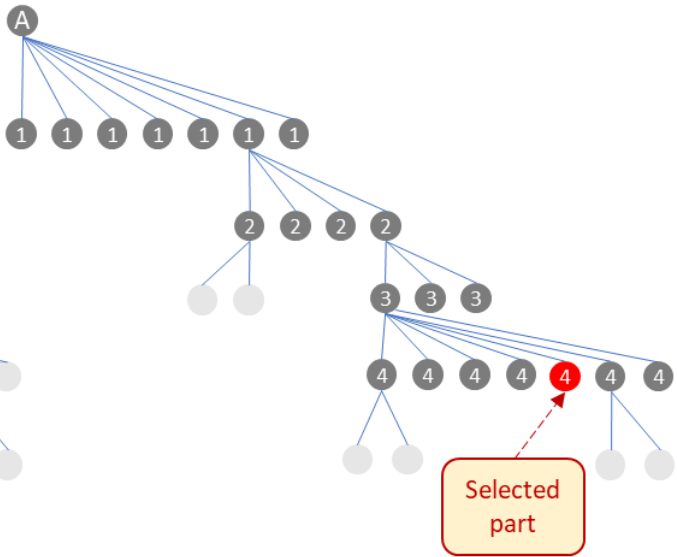
Figure 30 is used to illustrate the automated process of the system making subsequent queries to display the CAD Item rows in the Tree Grid View associated with the selected part. Assume the context CAD Item ('A') is shown in the Tree Grid View. The shaded circles in the diagram represent CAD Items and the numbers in each represent the query sequence number associated with displaying that CAD Item in the Tree Grid View. When the **Max Visible Children On Expand** is set to '3' the system makes 7 queries until it reaches the depth and breadth of the selected Path. Likewise, when the **Max Visible Children On Expand** is set to '10' the system makes 4 queries until it reaches the depth and breadth of the selected Path.

The **Max Grow Levels** field is used to set the depth of queries for each related Item. The default is 2. The larger the value, the deeper the query into each related Item. For a CAD Structure that is large (both in depth and breadth) these queries can take a long time before the Tree Grid View is updated.





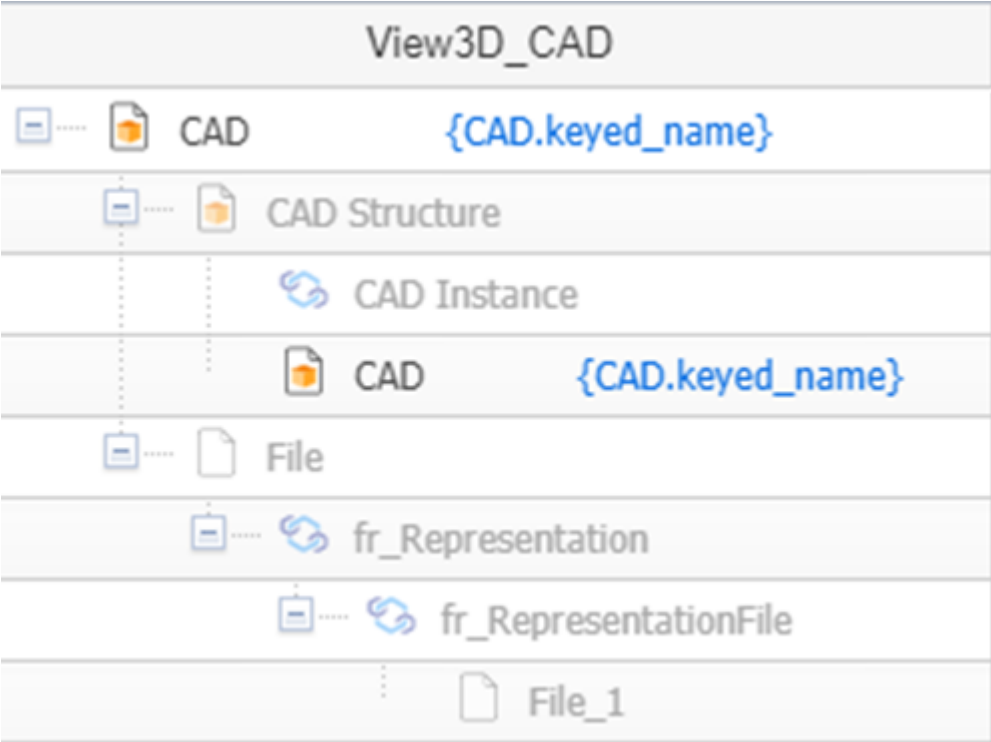
**CAD Item Query**  
(Max Visible Children On Expand = 3)



**CAD Item Query**  
(Max Visible Children On Expand = 10)

The default Tree Grid View simply maps the CAD Query Items from the Base Query and uses the keyed\_name Property for the node text.





## Customizing the Tree Grid View Definition

Use the following procedure to create a custom Tree Grid View Definition for use with the Dynamic and Streaming Viewers:

1. Create a new Tree Grid View Definition. It is necessary to create a new Tree Grid View Definition rather than copy the default Tree Grid View when a separate Query definition is being used.
2. Assign a **Name** and **Description**. They help identify the purpose of the Tree Grid View for future reference.
3. Choose a Query Definition based on the Base Query (see Section [Base Query Definition](#) ). Query Definitions must have the CAD ItemType as the context ItemType.

### Important

Once a Query Definition is selected and the Tree Grid View Definition saved, the selection of the Query Definition cannot be changed.

After saving the Tree Grid View Definition Item, the Editor View becomes available

4. Assign values for **Max Visible Children On Expand** and **Max Grow Levels**. Refer to section Default Tree Grid View Definition for a description of these values and their effect on the Dynamic and Streaming Viewers.
5. Assign the Query Definition Mappings. Refer to the Tree Grid View Administrator's Guide for a description of how to assign mappings to Query Items. Note that the name of the Tree Grid View Definition appears in the View Selection context menu.
6. Assign a Linked Toolbar/Context Menu. This is required for all Tree Grid Views associated with Dynamic View Definitions, created in the following step. If a Tree Grid View does not contain a Linked Toolbar/Context Menu, the user will receive an error. The default Linked Toolbar/Context Menu, **View3D\_CAD** Presentation Configuration, should be reused to maintain access to default Toolbar buttons and Tree Grid View Context Menu, described in section Context Menu Functions.

Create a Dynamic View Definition. The Dynamic View Definitions should be assigned to the specific viewer (Dynamic or Streaming Viewer)



## Dynamic View Definitions ▼ ☆

🔍 Search
🗑️ Clear
Simple ▼
Default \* ▼
📷 ▼
🖥️ ▼
🌐 ▼
⚙️

Target Item T...	Tree Grid V... ↑	Name ↑	View With [...]
<a href="#">CAD</a>	<a href="#">View3D_CAD</a>	Default Dynamic View Definition	<a href="#">Dynamic HOOPS Viewer</a>
<a href="#">CAD</a>	<a href="#">View3D_CAD</a>	Default Streaming View Definition	<a href="#">Streaming HOOPS Viewer</a>

7. Click on **New Dynamic View Definition**.

## Dynamic View Definitions ▼ ☆

🔍 Search
🗑️ Clear
Simple ▼
Default \* ▼
📷 ▼
🖥️ ▼
🌐 ▼
⚙️

Target Item T...	Tree Grid V... ↑	Name ↑	View With [...]
<a href="#">CAD</a>	<a href="#">View3D_CAD</a>	Default Dynamic View Definition	<a href="#">Dynamic HOOPS Viewer</a>
<a href="#">CAD</a>	<a href="#">View3D_CAD</a>	Default Streaming View Definition	<a href="#">Streaming HOOPS Viewer</a>

8. Enter the **Name** or the Dynamic View Definition Item which is used in the context menu for the Dynamic Viewer /Streaming as shown below.

9. Select the **Tree Grid View Definition**.

10. Select the viewer (**Dynamic** or Streaming) in **View With** field.

11. Assign a **Data Processor** method. "**dprn\_GraphAPIQueryProcessorMethod**" is available by **default** to process CAD.



# Filtering

Save Done Delete

## Dynamic View Definition

Name  
Filtering

Tree Grid View Definition  
View3D\_CAD

View With  
Streaming HOOPS View

## Data Processor

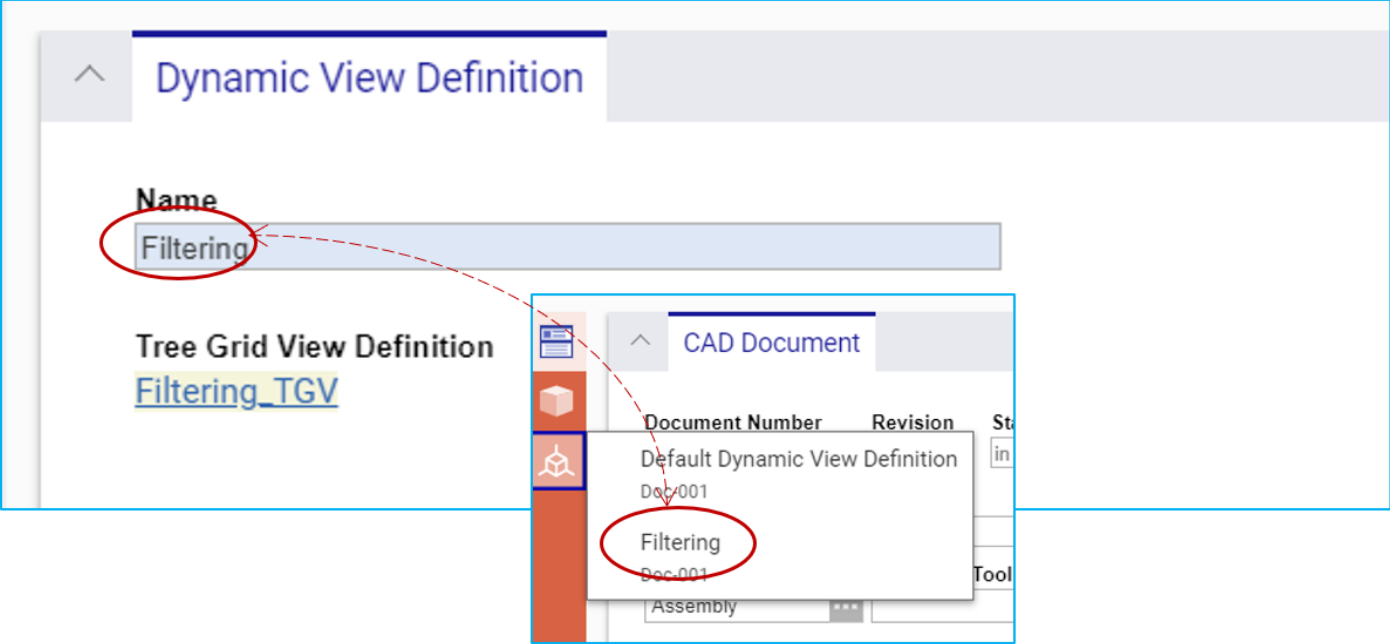
Methods

| Hidden |

Name ↑	Method T...	V..	execution_all...	Template [...]	Comments
dpn_GraphAPIQueryProce...	CSharp	1	World		

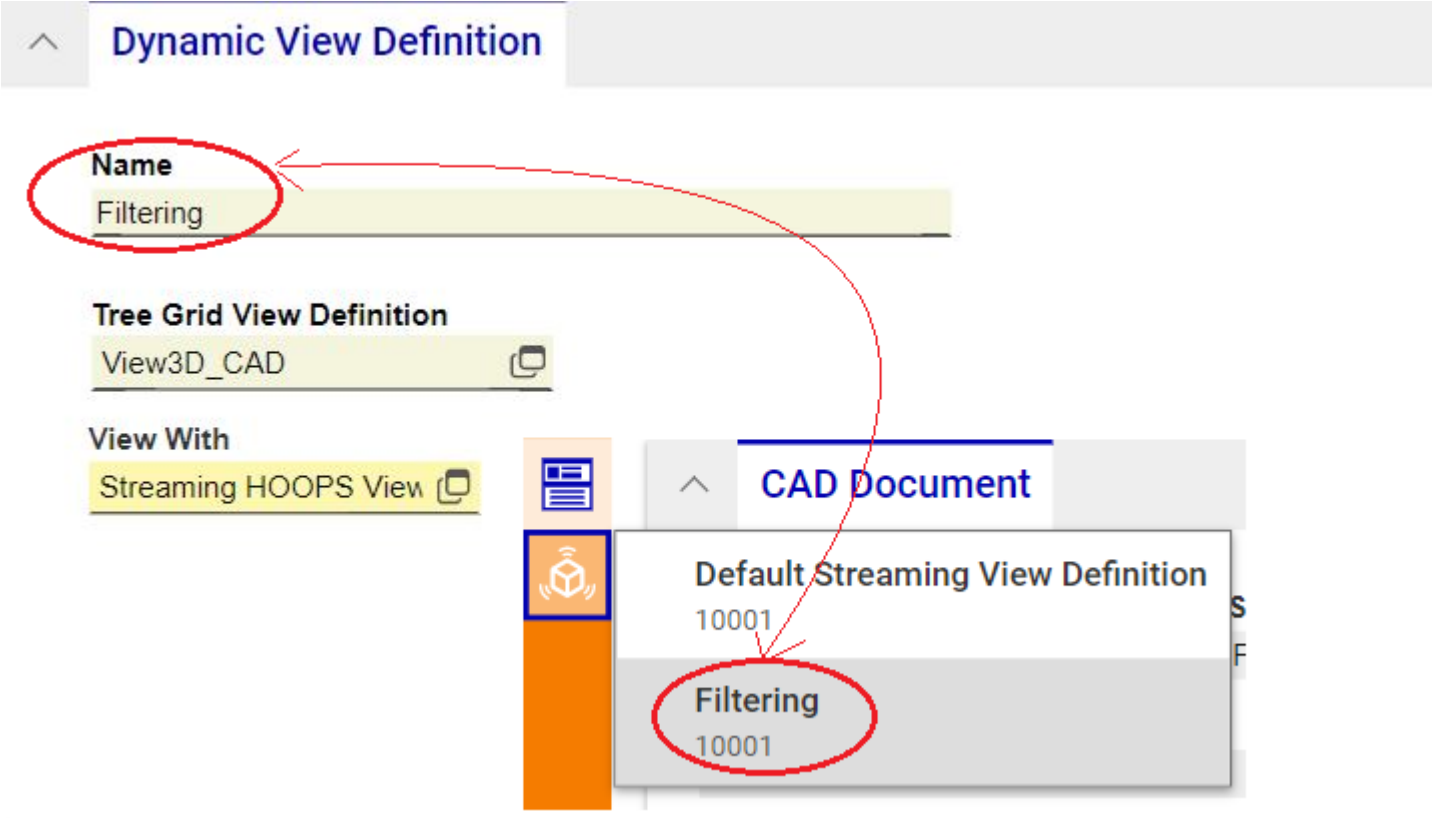
The Dynamic Definition for Dynamic Viewer appears as follows:





The Dynamic View Definition for Streaming Viewer appears as follows:





**Important**

The CAD Query Item must be mapped for the Tree Grid View to function properly in the Dynamic Viewer and Streaming Viewer.

Once set, the Tree Grid View Definition can be removed by removing the corresponding Dynamic View Definition Item.

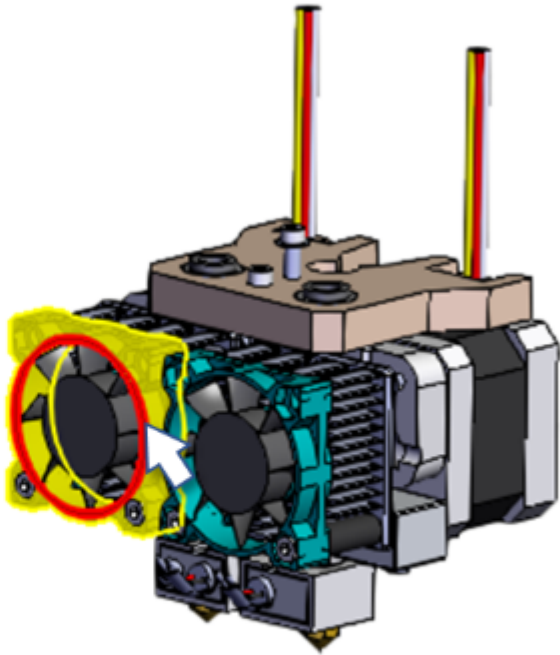


### Tree Grid View and 3D View Synchronization

Synchronization between the Tree Grid View and 3D View refers to the simultaneous selection of 3D component geometry and its corresponding Tree Grid View CAD Node. When selecting a CAD node in the Tree Grid View the corresponding 3D geometry is selected in the 3D View. Likewise, when a part is selected in the 3D View, the corresponding CAD node is selected in the Tree Grid View. Refer to Section Default Tree Grid View Definition for a description of the process of applying multiple queries until the selected CAD Item is 'uncovered' in the Tree Grid View.

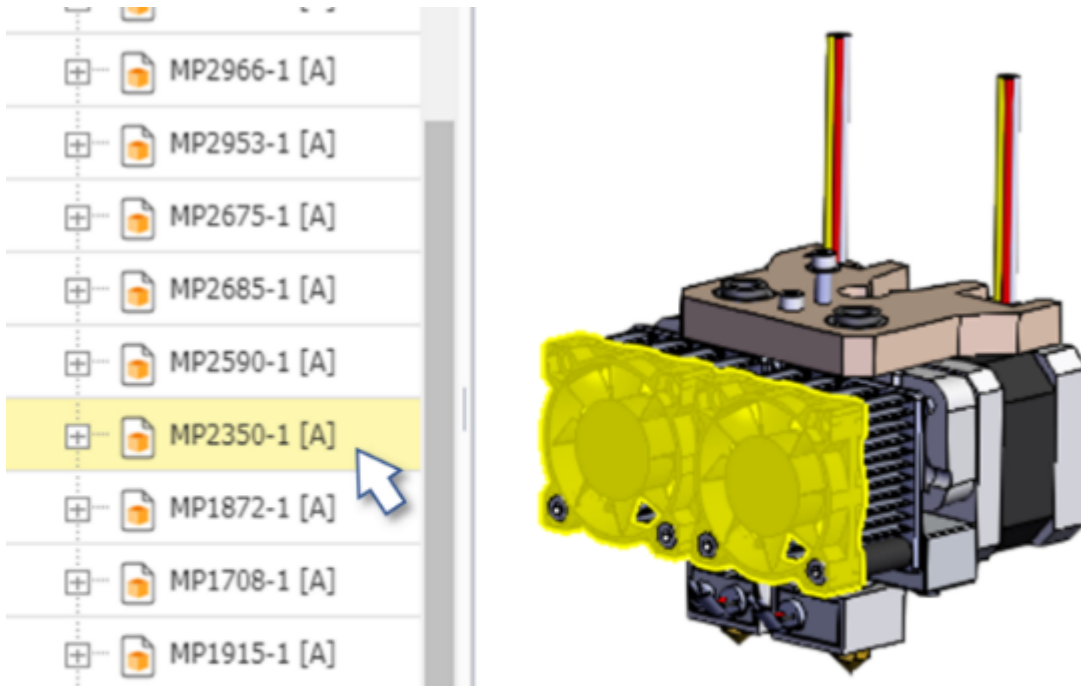
When a CAD Item Node is selected in the Tree Grid View, the 3D component geometry for all instances of the associated part is highlighted in the 3D View. The default Dynamic View Definition does not support the ability to select a single instance in the Tree Grid View. However, when selecting a part, only the 3D component geometry of the selected body of the part is highlighted. To configure the Tree Grid View to allow for selecting single instances, refer to section Creating a list of Product Occurrences.

- MP2966-1 [A]
- MP2953-1 [A]
- MP2675-1 [A]
- MP2685-1 [A]
- MP2590-1 [A]
- MP2350-1 [A]**
- MP1872-1 [A]
- MP1708-1 [A]
- MP1915-1 [A]



### Selection of 3D Geometry





Selection of Tree  
Grid View Node

## Selecting a Dynamic View Definition

### Important

The default DVD will be displayed as an option in the Dynamic Viewer/Streaming Viewer, along with any other configured DVDs for the context item type. If alternate DVDs are created, and the user no longer wants the default displayed, the default DVD must be removed by the root user.

To use a configured Dynamic View Definition (see Section Customizing the Tree Grid View Definition), select from the available choices when selecting (left click) the Dynamic View icon in the sidebar.

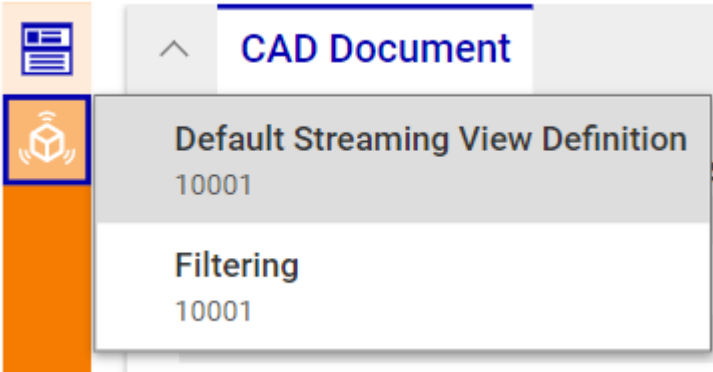
The Dynamic Definition for Dynamic **Viewer** appears as follows:

The screenshot displays the Aras Innovator web application interface. The main content area shows the configuration for a 'CAD Document' dynamic view definition. The configuration includes fields for Document Number (Doc-001), Revision, State (In Work), Assigned Creator, Designated User, From Template, and a list of checkboxes for 'Changes Pending', 'Standard', 'Template', and 'Dynamic Enabled' (which is checked). A 'Native File' field contains a link to 'Aras\_Cube\_27.S...'. Below the configuration, a 'Structure' tab is active, showing a table of CAD Documents.

Seque...	Document Nu...	Revi...	Name	Type	State	Native File [...]	Viewable File [...]	Authoring Tool	Dependency
128	Sample-00011	A	Part-Sample-00011	Mechanical/Part	In Work	<a href="#">Aras_Cube_1...</a>			
256	Sample-00011	A	Part-Sample-00011	Mechanical/Part	In Work	<a href="#">Aras_Cube_1...</a>			
384	Sample-00012	A	Part-Sample-00012	Mechanical/Part	In Work	<a href="#">Aras_Cube_1...</a>			

The Dynamic View Definition for Streaming Viewer appears as follows:





If there is a Dynamic View Definition that is currently in use, it will be highlighted.



## Auto-Execution

By default, users must refresh the view (by selecting the refresh button in the Tree Grid View toolbar) to load the 3D View and execute the initial query for the Tree Grid View (see Section Dynamic Viewer Process Overview) However, it is possible to execute the initial query when the 3D View is opened by selecting the **Execute Query On Opening Dynamic Viewer** option in the Secure Social Preference settings.

^ Preference

**Identity Id**  
[World](#)

^
General Settings
PE\_MainPreferences
Secure Social
Self-Service Rep

**Default bookmark**

**Default Number Of Replies**

**Default Number Of Flagged-By Users**

**Max Lines In Messages Before "More"**

Use Standard Toolbar For Viewers

Use Legacy 3D View Files

Execute Query On Opening Dynamic Viewer

**Message Notification**

Within the web client

Email on every message

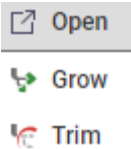
Email digest

**Prefix Text For Highlight Text Markup**



## Enabling the Open Context Menu Item

Tree nodes (rows) in the Tree Grid View have a default context menu with the **Open** menu item for opening the associated Item. This menu item is not enabled by default for the Tree Grid View Definition because the associated rows in the Tree Grid View Definition could be combined and thus reference more than one Item; see the Aras Innovator – Tree Grid View Administrator Guide for more information.



To enable the **Open** menu item to open the associated Item in the Tree Grid View:

1. Make sure that the `id` Property of the `ItemType` is included in the Query Definition (see Section Adding Properties). The `id` is required to identify the specific Item to open when the action associated with the View menu is executed.
2. Make sure the **Data Template** for the row in the Tree Grid View Definition includes the reference to the Item ID. The software that executes the **Open** function requires the `ItemType` and the ID of the specific Item to open the associated Form. For this, a data template is used by the Tree Grid View. This JSON template is defined within the Data Template for each row in the Tree Grid View Definition. The format of this JSON string is as follows:

a. `id` : ID of the Item to open

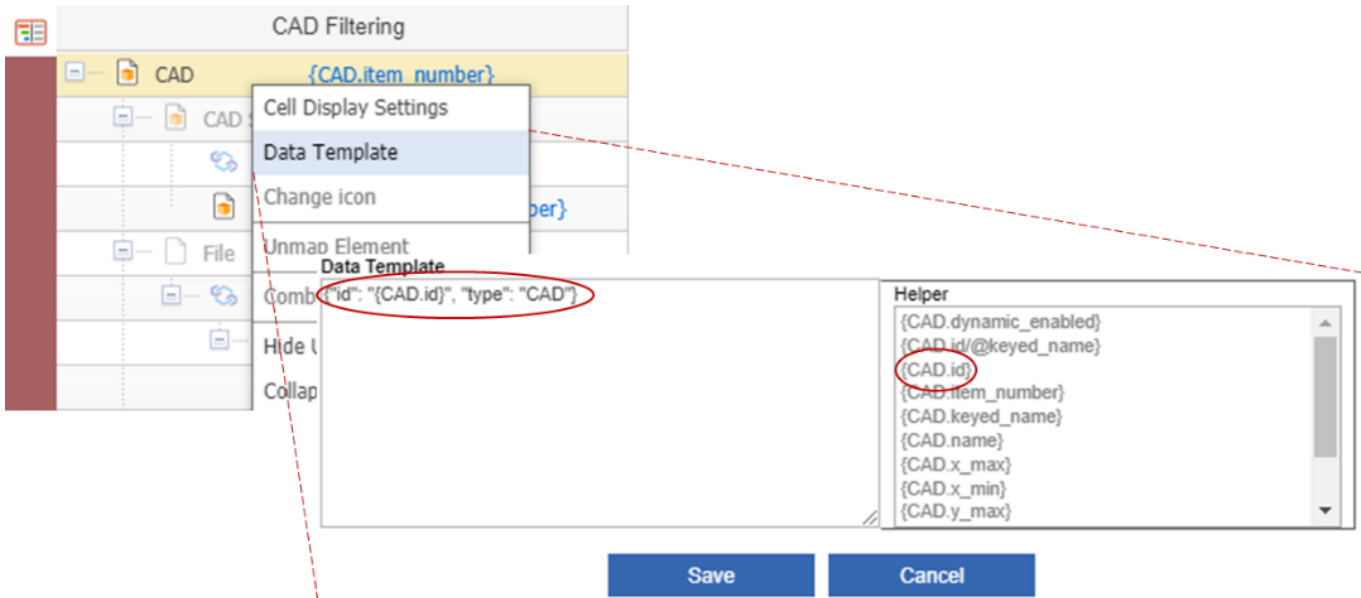
`type` : `ItemType` name of the Item to open

An example Data Template is:

```
{ "id": "{CAD.id}", "type": "CAD" }
```

Note the use of brackets - '{' and '}' - in the string. The brackets around the string `CAD.id` is used to extract the `id` of the Item associated with the row in the Tree Grid View when it is populated in the User Interface. All other strings are static.





**Creating a list of Product Occurrences:** It is possible to customize the Tree Grid View to support the display of instances by using the Tree Grid View ability to combine rows. The ability to synchronize the combined rows between the Tree Grid View and the Dynamic/Streaming Viewer relies on the reference path of the instance.



The following steps outline the process of configuring CAD for Instance Synchronization between the Tree Grid View and Dynamic/Streaming Viewer using the default Query and Query Processor:



## View3D\_CAD\_Instances

 Save  Done  Delete

^ Tree Grid View

<b>Name</b>	<b>Query Definition</b>
<input type="text" value="View3D_CAD_Instances"/>	<input type="text" value="View3D_CAD"/> 
<b>Context Item Type</b>	
<input type="text" value="CAD"/>	
<b>Description</b>	
<input style="width: 100%; height: 100px;" type="text" value="Definition for viewing CAD Instances"/>	
<b>Max Visible Children On Expand</b>	<b>Linked Toolbar/Context Menu</b>
<input type="text" value="100"/>	<input type="text" value="View3D_CAD Presentation Config"/> 
<b>Max Grow Levels</b>	
<input type="text" value="2"/>	
<b>Auto Grow On Refresh</b>	
<input type="checkbox"/>	

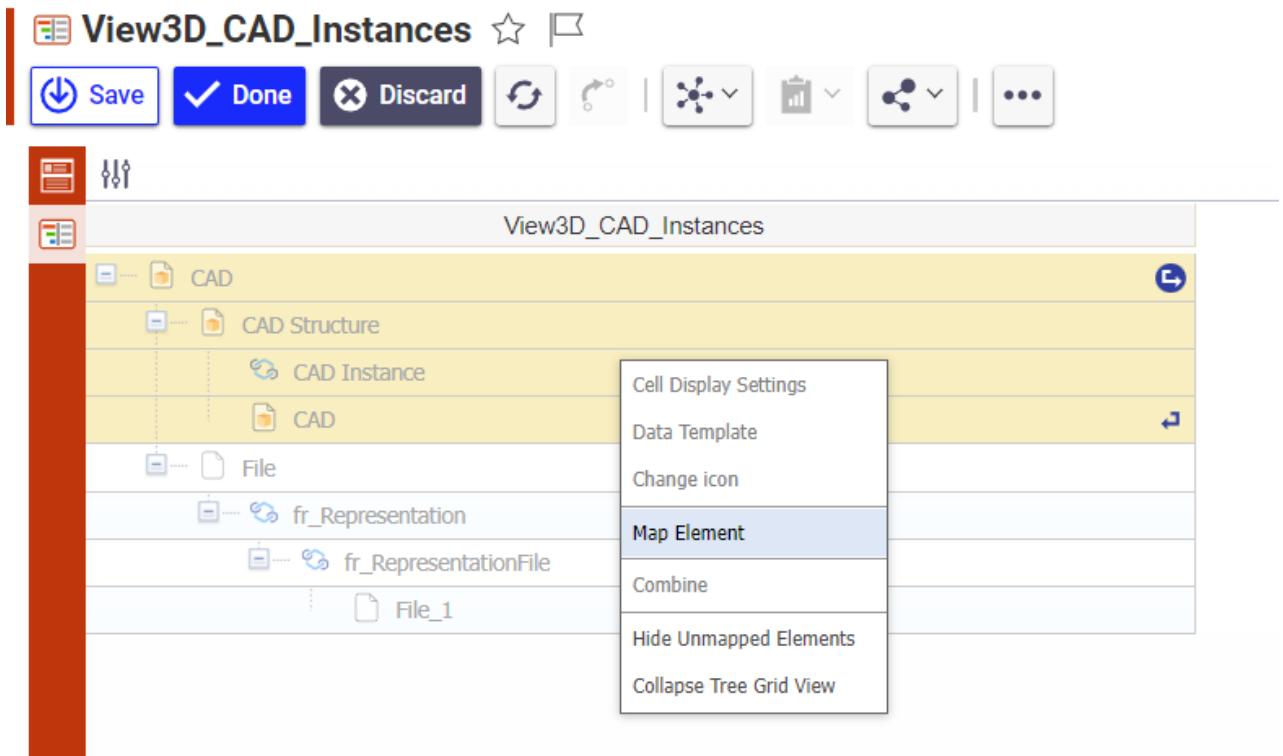
The following steps outline the process to display instances:

1. Create a Tree Grid View Definition:
  1. Go to the **Table of Contents** and select **Administration**.
  2. Select **Configuration**.
  3. Select Tree Grid View Definitions, and click Create New Tree Grid View Definition.
  4. Fill in the **Name** field.
  5. Fill in the **Description** field.
  6. Select the default View3D\_CAD Query Definition.
  7. Click **Save**.

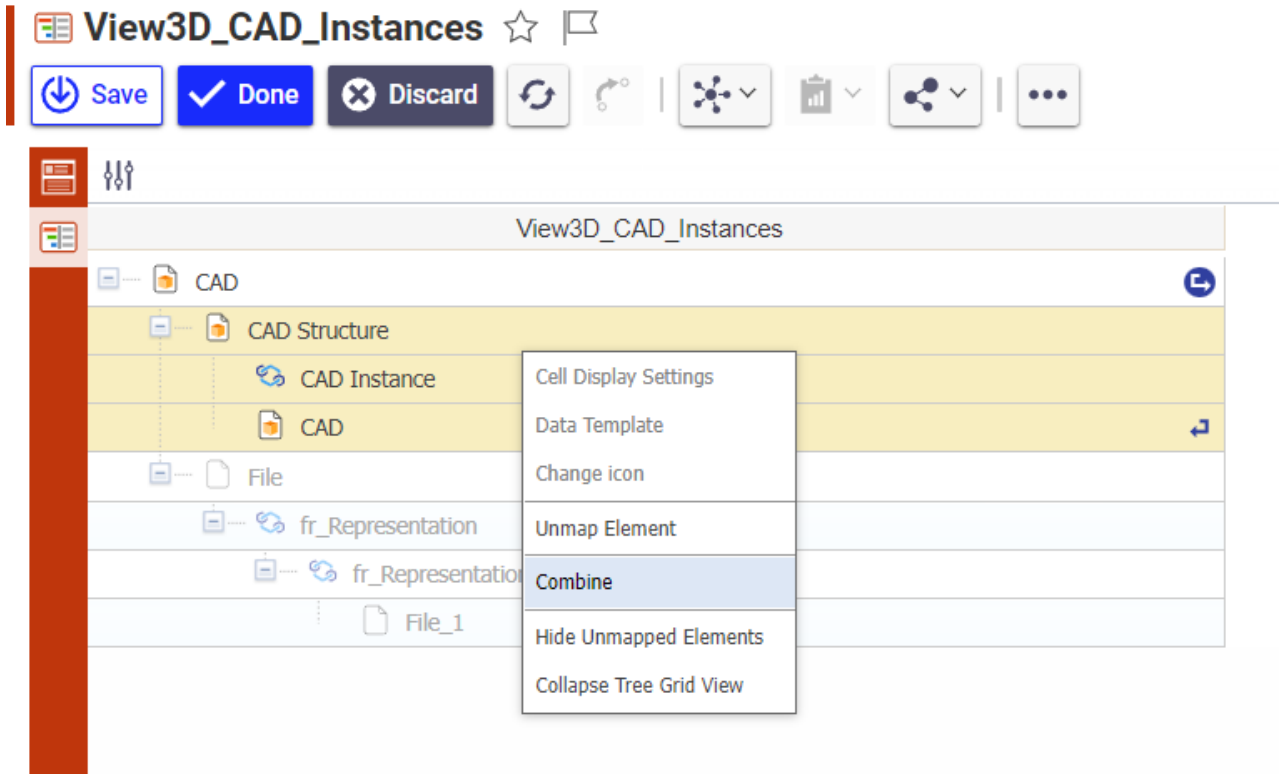


2. Modify the Tree Grid View:

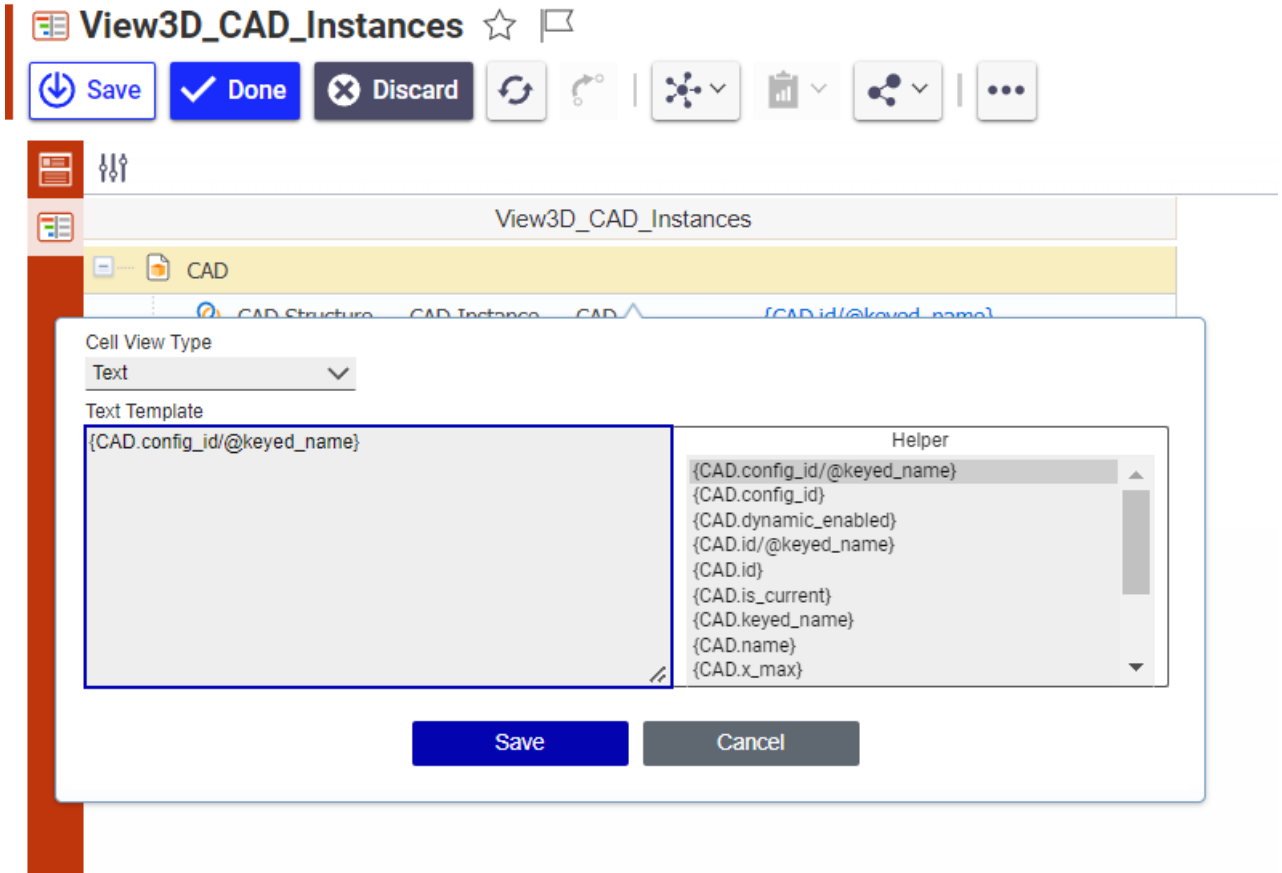
1. Click the Editor in the sidebar.
2. Select the first four rows (CAD, CAD Structure, CAD Instance CAD), right-click, and click Map.



3. Select the second, third, and fourth rows (**CAD Structure**, **CAD Instance**, **CAD**), right-click, and click **Combine**.



4. Select the first row (**CAD**), right-click, and click **Cell Display Settings**.



5. Select CAD.Keyed Name.
6. Select the second row (CAD Structure – CAD Instance – CAD), right-click, and click Cell Display **Settings**.



The screenshot shows the 'View3D\_CAD\_Instances' configuration window. At the top, there are buttons for 'Save', 'Done', 'Discard', and a refresh icon. Below these are icons for zooming and a table view icon. The main area displays a table with the following structure:

View3D_CAD_Instances	
CAD	{CAD.config_id/@keyed_name}
CAD Structure --- CAD Instance --- CAD	{CAD.id/@keyed_name}

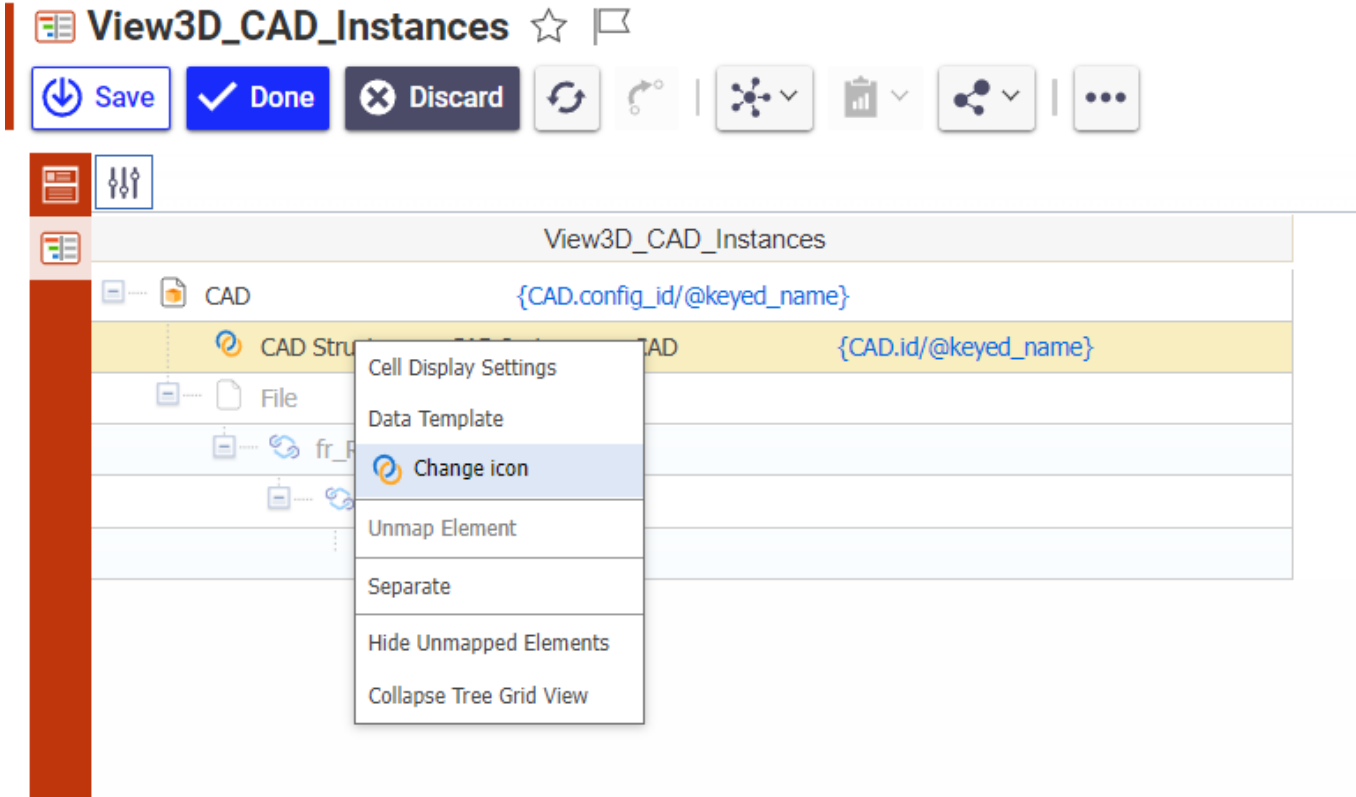
The 'Cell View Type' dialog is open, showing:

- Cell View Type: Text
- Text Template: {CAD.id/@keyed\_name}
- Helper list:
  - {CAD Instance.id/@keyed\_name}
  - {CAD Instance.id}
  - {CAD Instance.transformation\_matrix}
  - {CAD.config\_id/@keyed\_name}
  - {CAD.config\_id}
  - {CAD.dynamic\_enabled}
  - {CAD.id/@keyed\_name}
  - {CAD.id}
  - {CAD.is\_current}

Buttons for 'Save' and 'Cancel' are at the bottom of the dialog.

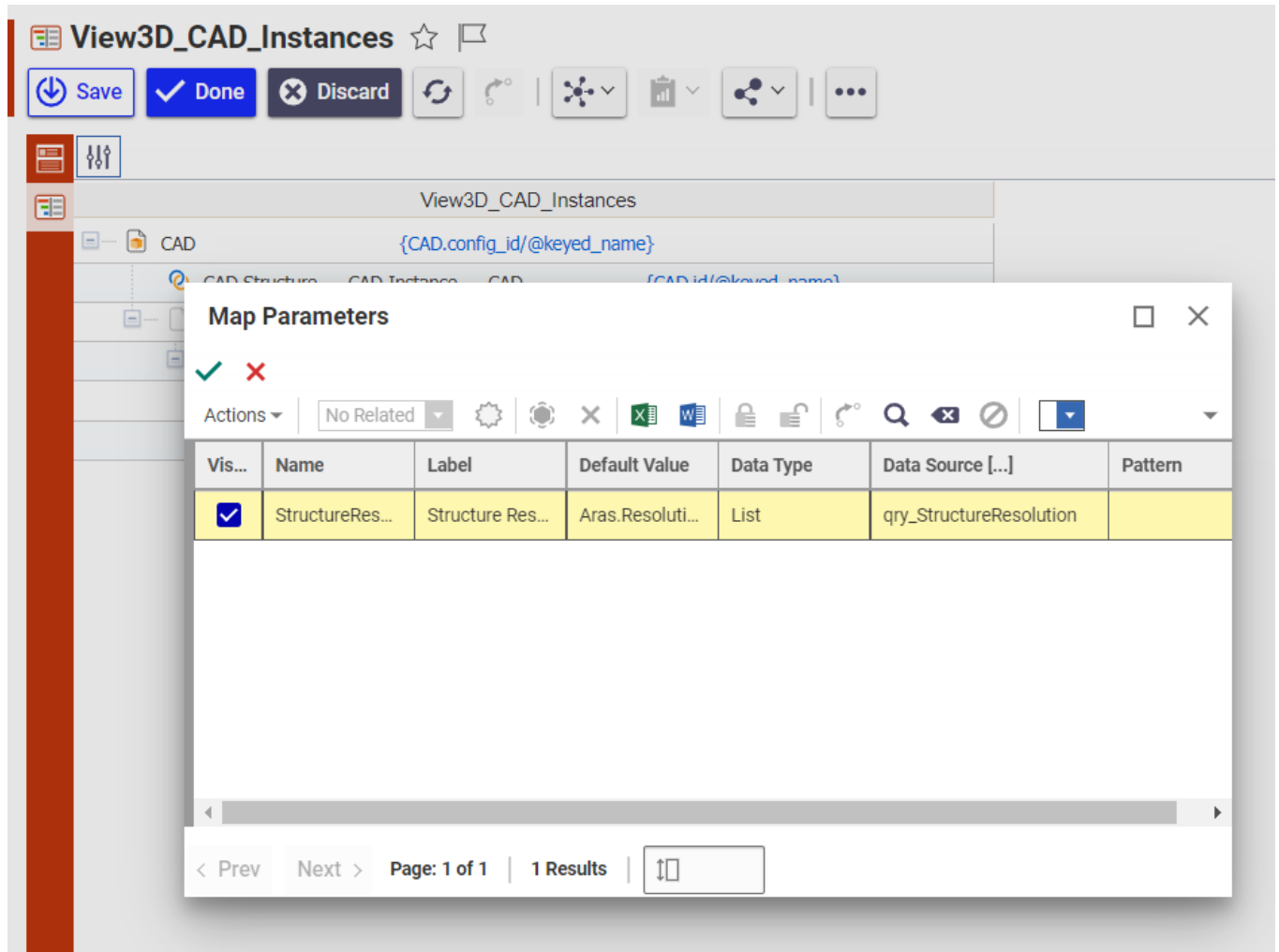
7. Select CAD.Keyed Name.
8. Optional: select an icon for the row.





3. Turn on Structure Resolution:





1. Click Map Parameters null
2. Select the Visibility check-box next to the StructureResolutionMode parameter.
3. Set Data Type to List.
4. Set Data Source to qry\_StructureResolution.
5. Click Save to close the Map Parameters dialog.
6. Click Done.

#### 4. Create a Dynamic View Definition:


1. Go to the Table of Contents and select Administration.
2. Select Configuration.
3. Click Dynamic View Definition and click Create New Dynamic View Definition.


## Instances Dynamic View Definition

 Save  Done  Delete



^ Dynamic View Definition









**Name**  
Instances Dynamic View Definition


**Tree Grid View Definition**  
View3D\_CAD 

**View With**  
Dynamic HOOPS Views 

^ Data Processor

 Methods 

  |   Hidden  |   

	Name ↑	Method T...	V..	execution_all...	Template [...]	Comments
	dpn_GraphAPIQueryProce...	CSharp	1	World		

4. Fill in the Name field.
  5. Select the Tree Grid View Definition created in Step 1.
  6. Select a specific viewer (Dynamic or Streaming Viewer) in View With field.
  7. Select the default dpn\_GraphAPIQueryProcessor data processor Method.
  8. Click Done.
5. Open a CAD Assembly and select the Dynamic View Definition created

MP0101 ☆

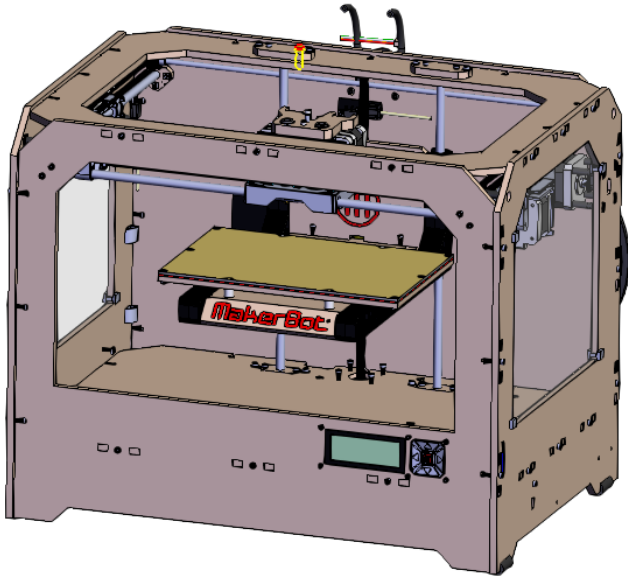


Model Views



View3D\_CAD\_Instances

- MP0101
- MP2942
- MP2968
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370
  - MP0370



## Alternate Query Processing

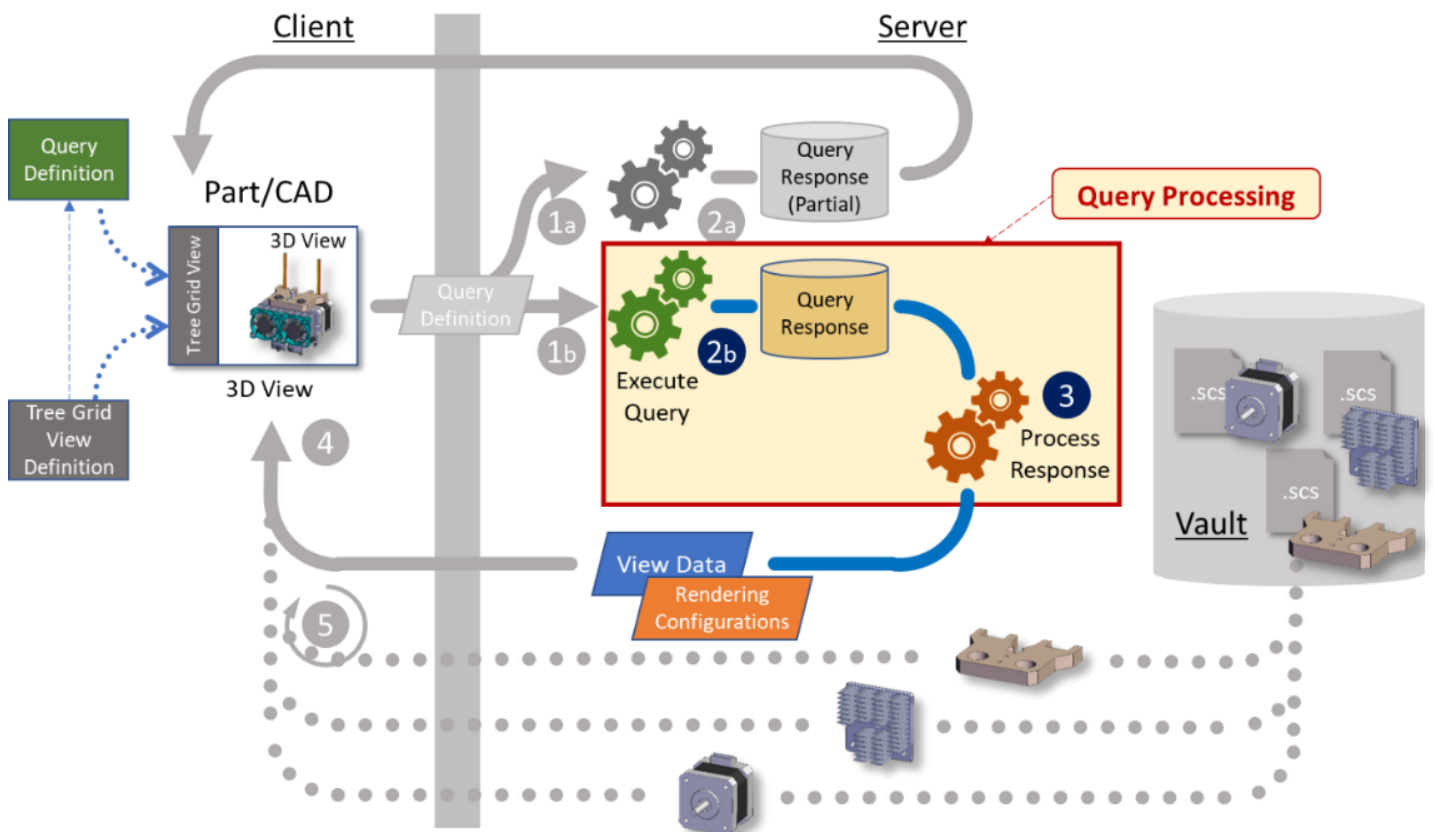
This section provides details for implementing and deploying custom Query Processors.



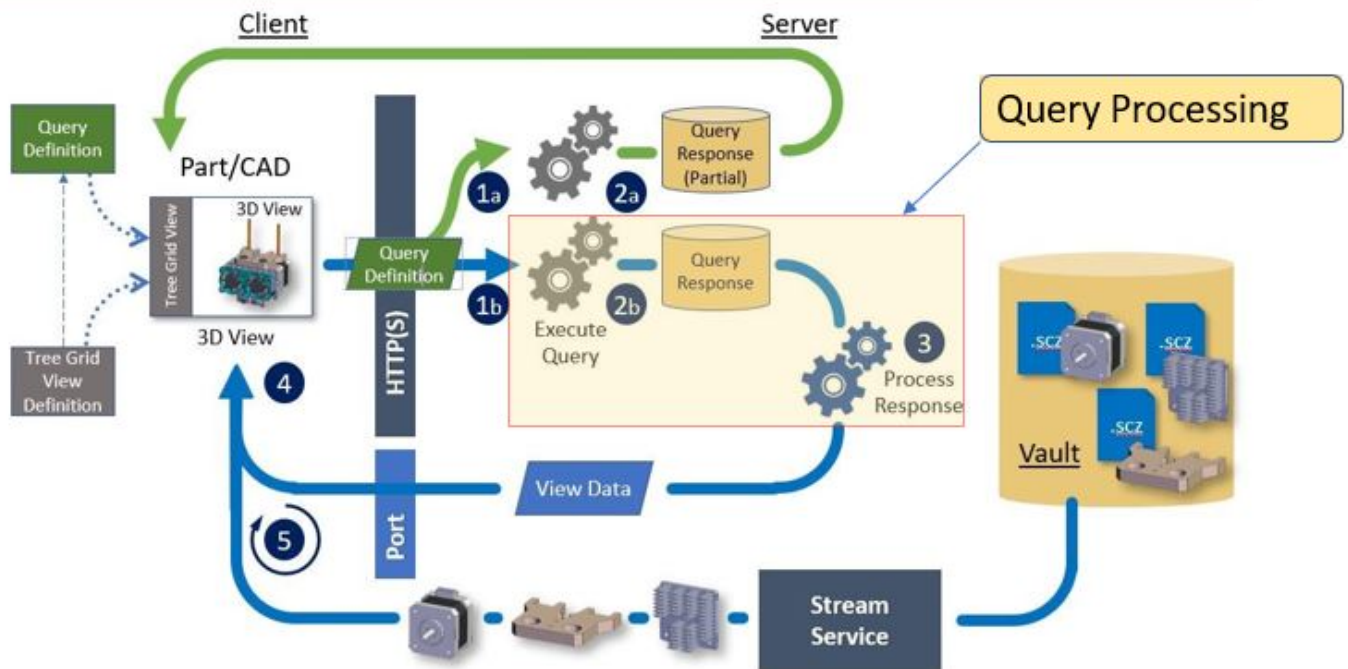
## Overview

For Dynamic Visualization, Query Processing refers to the process of executing a given Query Definition, parsing the results, and constructing a set of 'Product Occurrences' that represent the instances of the view geometry to display in the Dynamic or Streaming Viewer. Dynamic Visualization includes a Default Query Processor and, as explained in section *Error! Reference source not found.*, uses the CAD and CAD Structure data model as the source for the data required to construct a 3D View. As explained in section Creating Query and Tree Grid View Definition, this required data includes:

- 3D Component geometry
- Instances
- The transformations necessary to place and orient each instance



## Streaming Viewer Rendering Process (Shattered)



A Query Processing API was provided for Dynamic Visualization to allow implementations to utilize their own custom Query Processor when default processing will not fulfill the 3D Visualization requirements. A custom Query Processor provides the ability to:

1. Retrieve required data from alternate sources in a customer's data model.
2. Apply custom logic when processing the Query Definition execution results.
3. Create and apply alternate Rendering Configurations used to render the 3D View with different color and opacity for the 3D Component Geometry.
4. Map 3D Component Geometry to ItemType nodes displayed in the Tree Grid View other than CAD ItemTypes.

This capability greatly expands the applicability of 3D Visualization and enables use cases that are not possible using the Default Query Processor.

### Important

Query Processing for Streaming Viewers require SCZ files



## Implementing a Query Processor

### Important

Implementing a Query Processor requires knowledge of software development in .Net – specifically C#, IOM, XML and XML Processing, Query Definitions, and the data model (ItemTypes) used by the targeted implementation. As such, creating a custom Query Processor is a highly technical undertaking and should only be performed by competent users. This section is written assuming this level of knowledge.



## Create the Query Definition

Developing a custom Query Processor starts with the Query Definition. Users can use the default Query Definition as is, use a modified copy, or use the default simply as a reference when creating a new Query Definition. The specific ItemTypes chosen do not matter so long as the required data is either included in the Query results or is provided in some other manner within the logic of the Query Processor implementation. For this section the discussion will be referring to the Default Query Definition discussed in section Creating Query and Tree Grid View Definitions. It's important to note the following about this query:

- CAD as the context ItemType
- The Query is rooted (top level node) by a Query Item referring to the CAD ItemType.
- Recursion

There is a recursive relationship based on CAD and the CAD Structure Relationship.

CAD Structure defines the Mechanical Bill of Materials. Thus, there is a hierarchical relationship between upper Assemblies and lower sub-Assemblies and Components.

- Data required for custom processing logic
  - Instances – determined by the related CAD Instance Items with the Transformation strings included. CAD Instances identify both assembly and component instances of child CAD Items.
  - Bounding Box – Retrieved either from the CAD Item Properties directly (legacy CAD Items) or from the related CAD ConversionInfo.

### Important

The processing logic should check the CAD ConversionInfo Item and use those values if they exist. Otherwise, use the bounding box properties attached directly to the CAD Item.

- Transformations – attached to each CAD Instance Item. An Identity matrix is assumed when a CAD Instance does not exist. Also, the matrix ordering is assumed to be column centric. See section Base Query Definition. The transformation value is set during conversion and represents the format of the data expected by the HOOPS 3D Viewer. As such, it is used as it is stored when creating the Product Occurrence data as part of the output of a Query Processor.
- View Files – attached to each CAD Item through the native file Property. See section CAD Data Model.



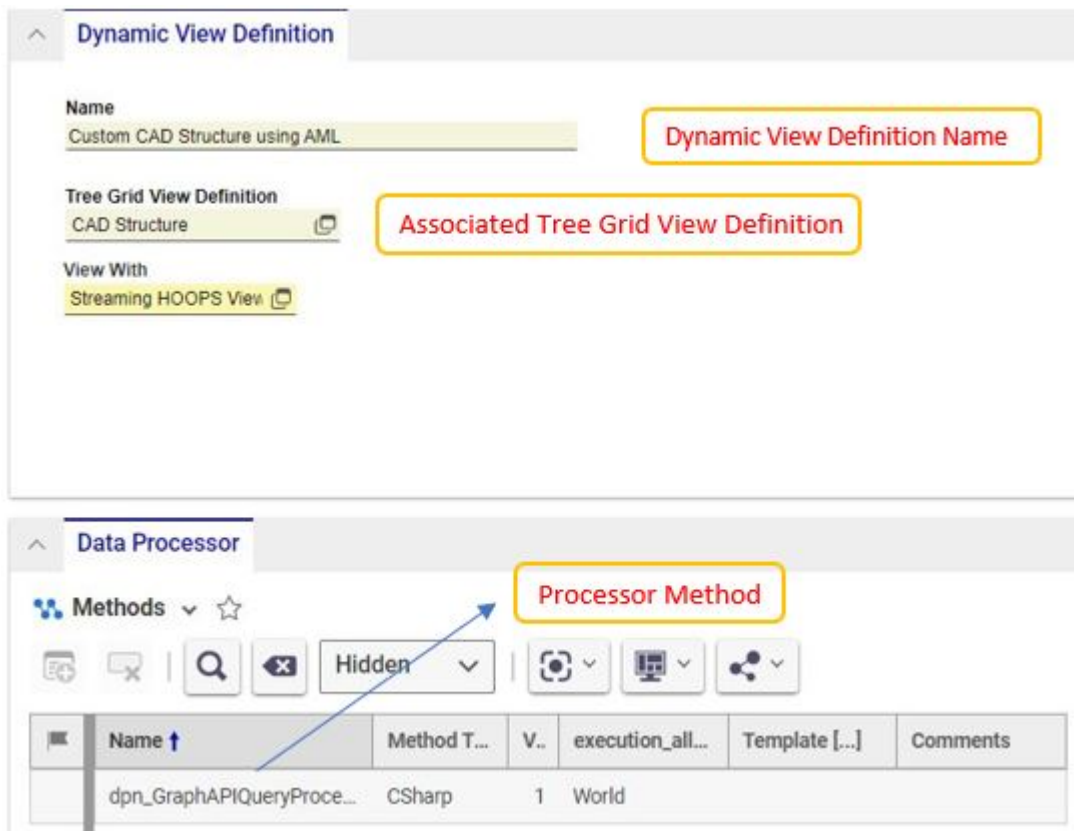
### Create the Tree Grid View

When creating a custom query processor, the Tree Grid View Definition can be defined as it is described in section [Tree Grid View Definitions](#) . No additional provisions need to be made other than ensuring that the Query Items used for mapping to 3D Component geometry are mapped and displayed in the Tree Grid View.



### Create the Dynamic View Definition with Data Processor Method

Dynamic View Definition Items are used to add the Dynamic or Streaming Viewer icon to the sidebar so the Dynamic or Streaming Viewer can be opened for an Item of a type associated with the configured Tree Grid View Definition Context ItemType. They can also be used to assign an alternate Query Processor for use with the Dynamic View Definition.



The execution of an alternate Query Processor starts with the assigned, server-side Method. This is referred to as the 'Data Processor Method'. Each Dynamic View Definition requires a 'Data Processor Method' to be assigned. The default Dynamic View Definition uses a default 'Data Processor Method' called by the **dpn\_GraphAPIQueryProcessor** Method.

#### Important



The Default Query Processor will only work with the Default Query Definition or with Query Definitions that use the same Base Query for context ItemType: CAD

The existence of a Dynamic View Definition Item triggers the application of the configured Tree Grid View Definition and its associated Query Definition as explained in prior sections. The following sections describe the execution of the Query Processor and how to create a custom implementation.

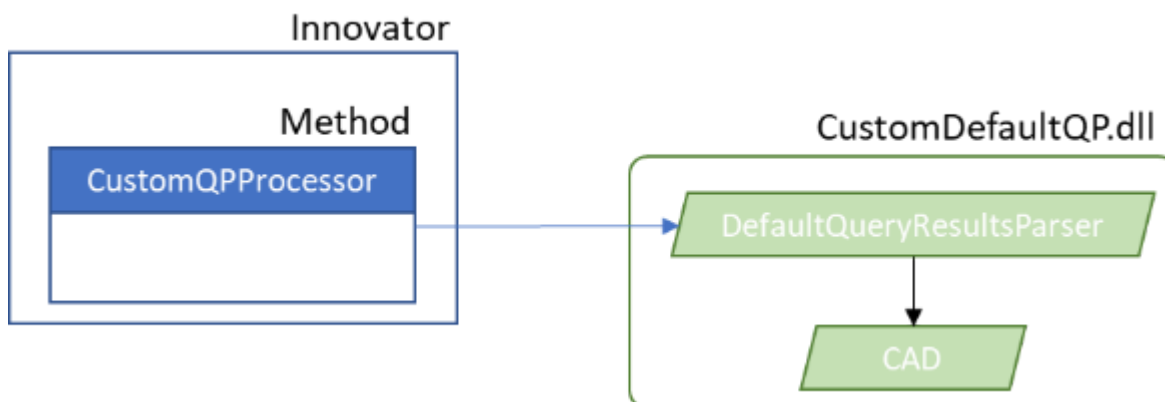


### Create the Data Processor Method

The example described in this section relegates the bulk of the Query Processing logic to a separate DLL which is linked with the Aras Innovator Server. This separation of code is not necessary, but because of the amount of software that may apply to custom query processing this method of implementation is easier to manage. The actual Data Processor Method therefore is much smaller and is used mostly to validate input and make the necessary calls into the custom DLL in this case.

#### Important

Note: It is recommended to make methods OS agnostic for use with Linux and Windows. The main incompatibility issues in operating systems that need to be considered when implementing the method are path case-sensitivity, path separators, OS-specific line-endings, OS-specific code. For more information about cross-platform development, see section 2.3 Cross-platform development in the Aras Innovator 31 - Programmer's Guide.



The following is an example of the CustomQPProcessor Method, which is used to execute the Default Query Definition, pass the results to a custom DLL for processing, and generate the view data used for the 3D Viewer – Product Occurrences. The example is described here:

```

1 //MethodTemplateName=Csharp:Aras.Public.Events.EventHandler<Aras.DynamicModelViewer.DataModel.DataProcessorArgs>;
2 If(eventArgs==null || !eventArgs.IsValid())
  return;
3 //Get Innovator Connection
  var connection = Innovator.getConnection();
4 //Get Query Definition Helper
  Aras.DynamicModelViewer.DataModel.Helpers.QueryDefinitionHelper qdHelper = eventArgs.QueryDefinitionHelper;
5 // Get Query Definition Item
  Item qdItem = qdHelper.GetQueryDefinitionItem(eventArgs.QueryDefinitionId);
6 // Apply Query Definition and get result
  Item qdResult = qdHelper.GetQueryDefinitionResultItem(qdItem, eventArgs.ItemId, eventArgs.QueryDefinitionParams);
7 //Custom Query Processing Class
  CustomDefaultQP.DefaultQueryResultsParser defQryResParser = new CustomDefaultQP.DefaultQueryResultsParser();
8 //Process the query response
  Var result = new Aras.DynamicModelViewer.DataModel.QueryProcessingResult();
  Result.SetProductOccurrenceList(defQryResParser.processQueryResults(qdResult));
9 //Return the results
  eventArgs.SetQueryProcessingResult(result);

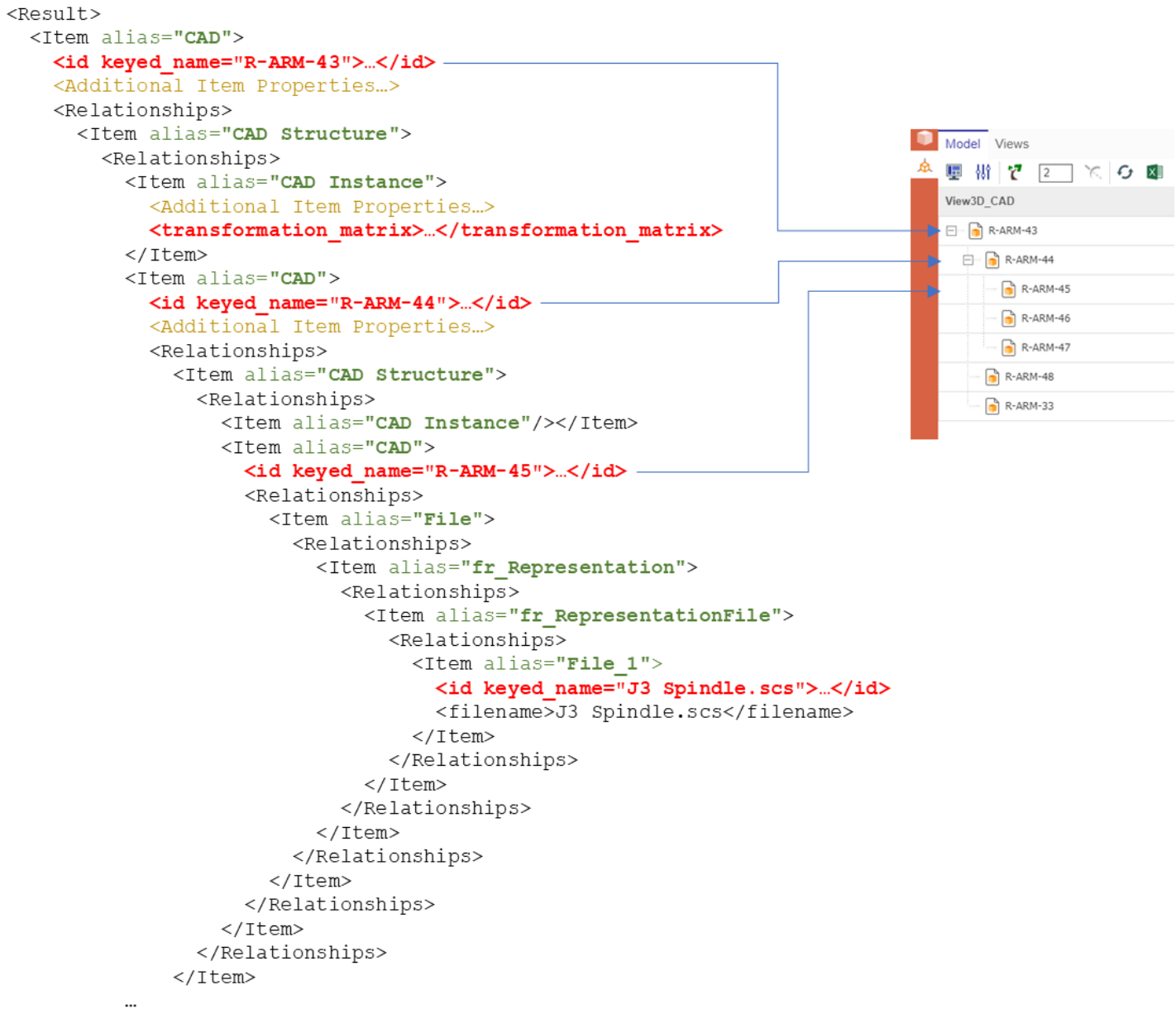
```

1. This initial comment must be the first line in the Method. It is used to identify the Method template to be used.
2. The EventArgs object is passed into, and available, to this Method. This object contains The Query Definition ID, ID of the Item being viewed, the Query Definition Parameters map, and will contain the results created by the Query Processor.
3. Retrieves the Aras Innovator connection.
4. Retrieves the Query Definition Helper used for this Query Processor.
5. Retrieves the Query Definition Item.
6. Execute the Query Definition using the given Item ID and the query parameters used.
7. This example includes a custom DLL used for the query processing logic, with the main class – CustomDefaultQP.DefaultQueryResultsParser. This software, explained below, is used to parse the query results, and construct the response used to populate the 3D Dynamic/Streaming Viewer.
8. The results of the processing – Product Occurrence List – is constructed by the custom DLL and stored in the QueryProcessingResults Object which is then returned to the EventArgs Object passed into the Method (see Help files for API documentation).

### Processing Results from Query Execution

The custom Query Processor example processes the results returned from the execution of the Query Definition. To illustrate this process this section will use the example query results shown in below figure and discuss each relevant XML Element when processing.





The above figure shows the partial XML result data when executing the Default (Base) Query Definition against a sample Assembly shown at the right. In this example, the root Assembly – R-ARM-43 – has one sub-Assembly – R-ARM-44 – and two child Components – R-ARM-48, R-ARM-33. Assembly R-ARM-44 has three Components – R-ARM-45, R-ARM-46, and R-ARM-47. The diagram highlights key XML Elements and Attributes in the XML results that are necessary when processing this data. The `<Additional Item Properties>` XML Element is notional: it



represents additional Properties included for the associated XML Element that were left out of the example for simplicity.

When implementing a Query Processor, it is necessary to process the result set such as displayed above and extract the required information. The following sections identify some points to consider for this purpose.

### **Aliases**

Each Item in the results is represented by an `alias` attribute matching the alias value provided in the Query Definition. This is important, because the alias identifies the specific Query Item that matches the associated Item (see section Base Query Definition). There can be multiple Query Items in a Query Definition that refer to the same ItemType. The alias is the only way to distinguish the resulting query information.

### **Data Model Object**

It is useful to collect the information extracted from the Query Results in a separate object, which can then be used to construct the Product Occurrence list. In the example shown in section, the Class `CAD` was included to store key information extracted from parsing the Query Results.



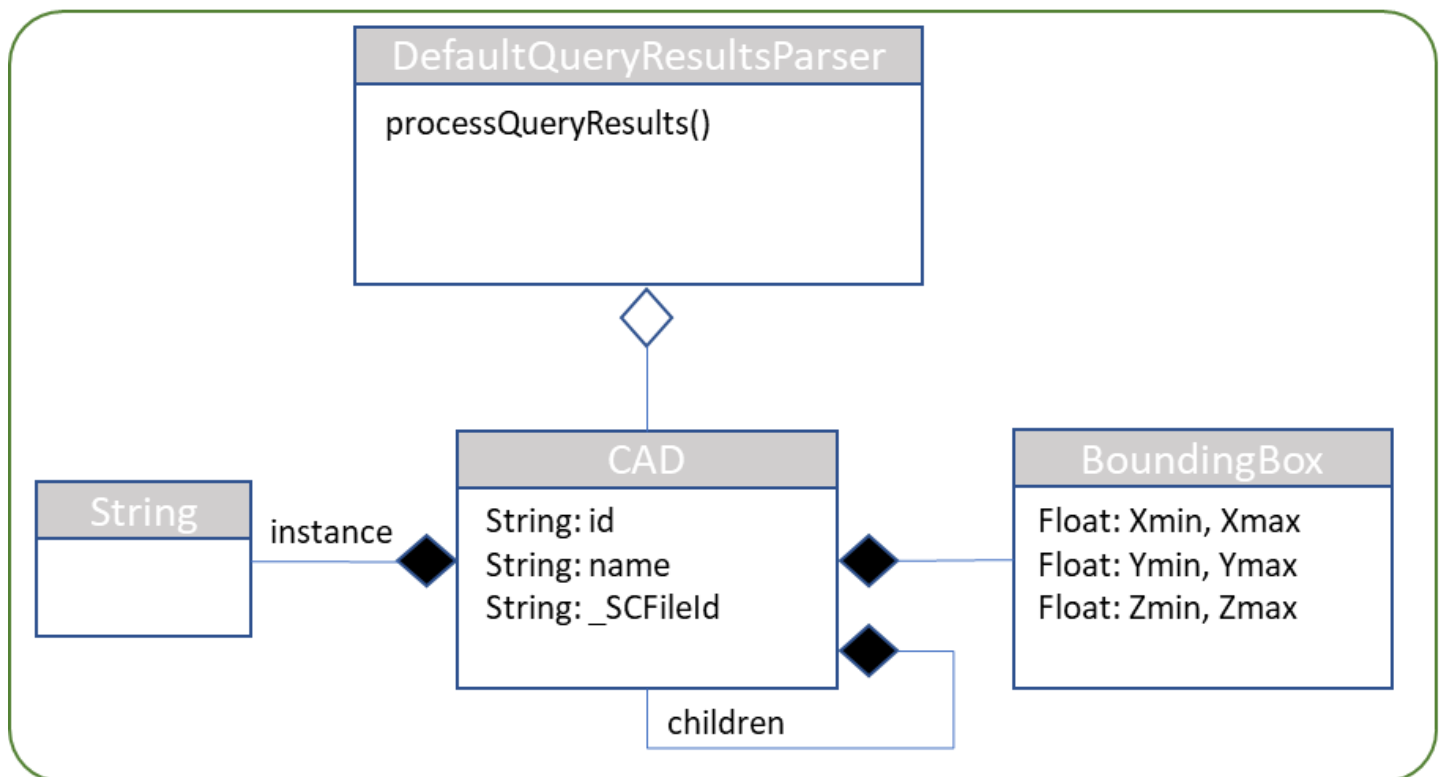
## Create the Query Processor DLL

This section describes an example set of classes that were created to process the results from the Default CAD Query. It is provided to show an example of how to process query results and generate the necessary list of Product Occurrences used by the 3D Viewer. The full set of example code is included in section Sample Custom Default Query Processor.

### Important

The code fragments shown in this section are for instructional purposes only. Users who create their own Query Processors should ensure the robustness, performance, and correctness of whatever is created to the environment it's meant to serve.

## CustomDefaultQP.dll



### DefaultQueryResultsParser

The class `DefaultQueryResultsParser` performs the two main functions of a Query Processor: Parse/Process the results from the execution of a Query Definition, create a list of



**ProductOccurrence** Objects representing the 3D geometry *View Data* to be displayed in the 3D Viewer.

### Processing Query Results

Processing the query results entails extracting the information necessary to construct the Product Occurrences as well as any additional information that will be used in constructing Rendering Configurations. At a minimum, the following is needed:

- ID and name of whatever element represents the 3D Geometry in the Tree Grid View.
- ID of the Query Reference that corresponds to the row in the Tree Grid View that the geometry will map to when selected in the viewer.
- BOM hierarchy of elements.
- Transformation matrices for each Instance of 3D Geometry.
- ID of the view file used for each 3D geometry component.

#### Important

Legacy environments used the CAD ItemType to store data from the CAD Conversion process. The **CAD ConversionInfo** Relationship was added to obviate the need to update a CAD Item (and thus lock it) during the conversion process.

In this example, the element representing 3D Geometry is the CAD Item. Note that this could be a Part Item, or some other custom Item used in the data model. It represents the node that is selected in the Tree Grid View when the 3D component geometry is selected in the 3D View. The hierarchy of 3D Components is also important since transformations are applied top-down. In the default CAD Data model in Aras Innovator, Child CAD Items are transformed relative to the transformation applied to their parent CAD Item, and so on. Transformations are required to position the instance of the 3D geometry in 3D space. Finally, the ID of the view file uniquely identifies the view file in the Aras Innovator Vault.

The Default CAD Query Definition will return one or more root CAD Items, which represents the CAD Item the Query Definition was executed against along with any additional CAD Items selected using Digital Mockup (see Section ). Each of these CAD Items should contain the same Properties so processing the full CAD Hierarchy can be done recursively. For each CAD Item parsed from the Query Execution results, a single CAD Item Data Object is created.



In this example, and in Figure 54, the CAD Class is used to process the data from the Query Execution results and store the data identified above. For each instance parsed, the transformation string is extracted and stored as a String within a list. The number of transformation strings in this list represent the number of instances of the current CAD Item relative to its parent CAD Item. Bounding box data, if present, is collected in a simple struct with double attributes for each X/Y/Z min and max values.

### Important

Adding bounding box data to the Query Definition and the generated Product Occurrences should be included to help position the camera when the model is initially rendered.

It is recommended to make methods OS agnostic for use with Linux and Windows. The main incompatibility issues in operating systems that need to be considered when implementing the method are path case-sensitivity, path separators, OS-specific line-endings, OS-specific code. For more information about cross-platform development, see section 2.3 Cross-platform development in the *Aras Innovator 31 - Programmer's Guide*.

If a given CAD Item is an assembly, it should contain child CAD Items. In this case, each of those CAD items are processed recursively down to the component CAD Item. Component CAD Items need to parse the associated view file data. Only view files for Component (NOT Assembly) CAD Items should be included in the Product Occurrences. The following is a code snippet showing a possible implementation of processing CAD Item Data:

```
internal void processCAD(QueryBuilderNode qbItem)
{
    QryRefId = qbItem.QueryItem.RefId; // Query Item Reference ID
    try
    {
        ID = qbItem.GetProperty("id"); // required
        Name = qbItem.GetProperty("name"); // required
    }
}
```



```
catch
{
    throw new ArgumentException("Query Definition is not defined properly:
'id' and 'name' Properties are required for CAD Items");
}

// Bounding Box Data. Alternate values ensure that there is a non-empty
// bounding volume defined

// NOTE: Bounding box data should first be retrieved from the related CAD
ConversionInfo Relationship. If not present, retrieve from the CAD Item
directly

_BBox.MinX = _getPropertyAsDouble(qbItem, "x_min", -1.0);
_BBox.MaxX = _getPropertyAsDouble(qbItem, "x_max", 1.0);
_BBox.MinY = _getPropertyAsDouble(qbItem, "y_min", -1.0);
_BBox.MaxY = _getPropertyAsDouble(qbItem, "y_max", 1.0);
_BBox.MinZ = _getPropertyAsDouble(qbItem, "z_min", -1.0);
_BBox.MaxZ = _getPropertyAsDouble(qbItem, "z_max", 1.0);

// for processing CAD children and associated view file
foreach (var child in qbItem.ChildNodes)
{
```



```

if (child.QueryItem.Alias == "CAD ConversionInfo")
    _processCADConversionInfo(child);
if (child.QueryItem.Alias == "CAD Structure")
    _processCADStructure(child);
if (child.QueryItem.Alias == "File")
    _extractFileInfo(child);
} // foreach
} // processCAD(QueryBuilderNode qbItem)

```

In this sample, the method assumes the type of Query Node provided refers to a `CAD` ItemType. In the Default Query Definition, CAD Items include Properties for each `Property` and Child Nodes for related `Files` and `CAD Structure`. Note that the Properties for 'id' and 'name' are required and an exception is thrown if they are not found.

#### Creating a list of Product Occurrences

Creating a list of Product Occurrences entails processing the CAD Data Objects created when parsing the query results. Note that although the API requests a *list* of Product Occurrence objects containing only the root Product Occurrence Objects representing the CAD Items being viewed at the top level – and each Product Occurrence will store the BOM hierarchy as *children*. The following is a code snippet showing a possible implementation for creating Product Occurrences.

```

private void build(ProductOccurrenceInstance parent, CAD c)
{
    foreach(String xFormStr in c.Instances)
    {
        ProductOccurrenceInstance poInst = new ProductOccurrenceInstance();
    }
}

```



```
//poInst.Attributes.Add(new ProductOccurrenceAttr("QUERY ITEM REF ID",  
c.QryRefId));  
  
//poInst.Attributes.Add(new ProductOccurrenceAttr("ITEM ID", c.ID));  
  
poInst.SetServiceProperty("QUERY ITEM REF ID", c.QryRefId);  
  
poInst.SetServiceProperty("ITEM ID", c.ID);  
  
  
// bounding box  
  
poInst.ProductOccurrenceSource.BoundingBox.Max.X = c.BBox.MaxX;  
  
...  
  
// Testing Rendering Configuration  
  
ProductOccurrenceRenderingConfiguration rConfig = new ProductOcc...();  
  
rConfig.SetColor(Color.FromArgb(55, 40, 0));  
  
rConfig.Opacity = 0.4;  
  
rConfig.Name = "test";  
  
poInst.RenderingConfigurations.Add(rConfig);  
  
  
// Add Transformation Matrix  
  
poInst.TransformationMatrix = cadInstInfo.XForm;
```



```
// Add SetServiceProperty Method Calls
poInst.SetServiceProperty(cadInstInfo.RefID, cadInstInfo.ID);
poInst.SetServiceProperty(c.CADStrQryRefId, c.CADStructureID);
poInst.SetServiceProperty(c.QryRefId, c.ID);
// Add SetServiceProperty Method Calls

poInst.Name = c.Name;
if (c.Children.Count == 0 && c.SCFileID != null) // Assume this is a
component
{
poInst.ProductOccurrenceSource.Name = c.SCFileID; // use File Item Id for
name
poInst.ProductOccurrenceSource.FileReferenceByTypeMap.Add(
SourceModelType.Scs, c.SCFileID);
}

if (c.Children.Count == 0 && c.SCFileID == null)
poInst = null; // Invalid Part
else (parent != null)
parent.Children.Add(poInst);
```



```
// Recurse through child hierarchy
foreach (CAD child in c.Children)
    build(poInst, child);
} // foreach(String xFormStr in c.Instances)
} // build()
```

This logic is complex, so each key section in the code is enumerated with an explanation that follows:

1. This example uses a recursive method to construct the hierarchy of Product Occurrence Objects. Note that the specific Product Occurrence classes used will either be `ProductOccurrenceInstance` or `ProductOccurrenceSource`. The former refers to actual instances (Assembly or Component) of some 3D geometry, the latter refers to the view geometry file itself.
2. As mentioned previously in the explanation of the CAD Data Model Object class, the list of transformation strings associated with each CAD object denote the number of instances for that CAD Item. There should be at least one.
3. For each Product Occurrence Instance, it's critical that the following two Attributes be included:
  1. Query Item Reference ID. The reference ID is used for all Product Occurrence Instances and is extracted from the Query Results
  2. ID of the Item. This is used to map the instance of the geometry to the node of the CAD Item displayed in the Tree Grid View.

#### Important

In 3DV 14.0.4, logic for associating Product Occurrence objects with Tree Grid Views (TGV) was modified. This change now uses the `SetServiceProperty()` method for setting the Query Item Reference ID and Item ID values. Previous versions used Product Occurrence Attributes for this purpose. In order to maintain synchronization between the 3D View and TGV for selection, use of `SetServiceProperty()` is necessary as shown in the example above.

#### Important

For each Product Occurrence Instance at the root level, it is necessary to set the "Root Flag" property set by calling `ProductOccurrenceBase.SetAsRoot()` method. In the case of Digital Mockup (section Digital Mockup), there may be several root Product Occurrence Instances. See the full example in Section Sample Custom Default Query Processor.



4. Set the bounding box values for each of the min and max values for X, Y, and Z
5. Add a Rendering Configuration. This is optional, but necessary to add one or more View Modes to the Dynamic or Streaming Viewer (section *Rendering Configurations*). Rendering Configurations add alternate rendering parameters for the geometry associated with the Product Occurrence. This includes color and opacity (transparency). Together they can be used to visually isolate/highlight certain 3D geometry to show some information about the model or about related Items. There can be multiple rendering configurations for each Product Occurrence, although each must have a unique Name.
6. If a Transformation string is included in the CAD Instance Item, then it should be applied to the TransformationMatrix Property of the Product Occurrence. If there is only one instance, and the geometry can be rendered as it was positioned/stored in the CAD software, then a transformation matrix is not required. In this case, an Identity matrix will be assumed. Note that if there are multiple instances of a CAD Item, then there should be a transformation matrix for each of them. Without a transformation matrix for each, the geometry will be rendered at a location based on how the geometry was stored in the CAD system.
7. Set the Instance properties for instance id, CAD id, and CAD Structure id to support combined rows and the synchronization of instances. This is required to synchronize combined rows in the TGV to extract the path of the combined row and map it to the view.

### Important

For each Product Occurrence Instance, it is necessary to set the service properties `SetServiceProperty()` method. See the full example in Section Sample Custom Default Query Processor.

8. The name used for the Product Occurrence. It's helpful to either use the document number or name of the CAD Item.
9. For the CAD Data Model Object used, if there are no children, then it is assumed that the CAD Item is a component; in which case it should have an associated view file. In this case, create a `ProductOccurrenceSource` object, setting the properties as shown. Note that the 'Name' needs to store the File ID of the view file.
10. If there are no children (CAD is assumed to be a Component) and there is no associated view file, then there is no sense (and it is invalid) in creating a Product Occurrence since there is no 3D geometry to load.

### Important

Errors returned from the processing of Product Occurrence lists regarding the value 'null' for 'key' Parameters are typically related to the inclusion of Product Occurrence Instances for assemblies where no child in that assembly contains associated view data. To avoid these errors, ensure that at least one descendant Product Occurrence has a valid view file attached to it.

11. If a valid parent was passed into the method, then add the newly created Occurrence to it.
12. Recurse through the CAD hierarchy.

## Product Occurrence List – View Data

The list of Product Occurrence objects generated by the Query Processor result in an XML set that is like the following example:



```

<Root assyId="...">
  <ModelFile>
    <ProductOccurrence Id="0" Name="R-ARM-43" Children="2 10 12">
      <Attributes>
        <Attr Name="QUERY ITEM REF ID" Type="String" Value="..." />
        <Attr Name="ITEM ID" Type="String" Value="..." />
        ...
      </Attributes>
      <Transformation RelativeTransfo="..." />
    </ProductOccurrence>
    <ProductOccurrence Id="2" Name="R-ARM-44" Children="4 6 8">
      <Attributes>
        ...
      </Attributes>
      <Transformation RelativeTransfo="..." />
    </ProductOccurrence>
    <ProductOccurrence Id="4" Name="R-ARM-45" InstanceRef="3">
      <Transformation RelativeTransfo="..." />
    </ProductOccurrence>
    <ProductOccurrence Id="3" Name="R-ARM-45">
      <ExternalModel Name="107FC69D133C4CBF8B1B90C5AF2E2E30">
        <BoundingBox Max="..." Min="..." />
      </ExternalModel>
    </ProductOccurrence>
    <ProductOccurrence Id="6" Name="R-ARM-46" InstanceRef="5">
      <Transformation RelativeTransfo="..." />
    </ProductOccurrence>
    ...
  </ModelFile>
</Root>

```

Assembly Instance

Component Instance

File Source Instance

Note the following about the XML View Data:

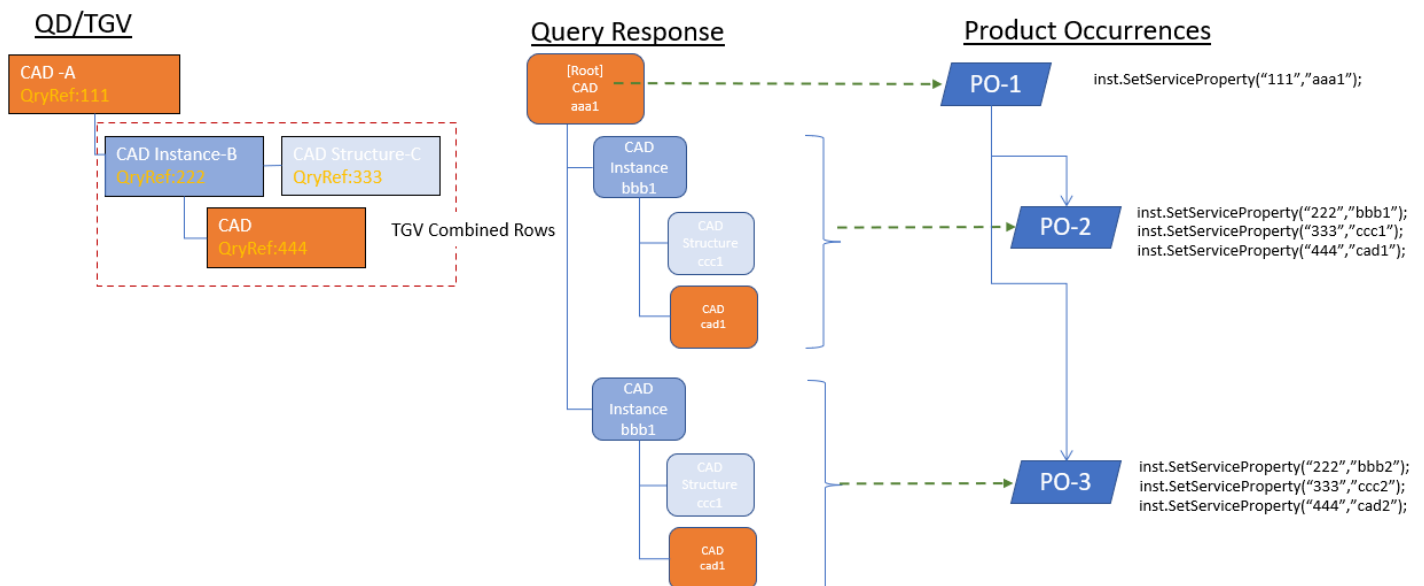
- The format of the XML is specific to the HOOPS Viewer; the XML schema is defined by HOOPS. It is generated automatically from the generated Product Occurrence objects.
- < ProductOccurrence > (one r) XML Elements define both Instance and File source information. This is consistent with the two types of Product Occurrence objects defined above when generating the Product Occurrence list.
- The data is flat, in that the hierarchy is defined by the Children attribute.
- All IDs are generated automatically and must be >=0. Each ID uniquely identifies the Product Occurrence.
- Attribute Elements are used for mapping instances of the 3D geometry to the nodes in the Tree Grid View. Note the Query Item Reference and ID are set as defined in section Creating a list of Product Occurrences.
- Component Instances reference their geometry using the InstanceRef attribute.
- The system will ensure that there are no duplicate File Source Instances. Thus, reused geometry will 'point to' the same Product Occurrence for the File Source.



## Processing Combined Rows

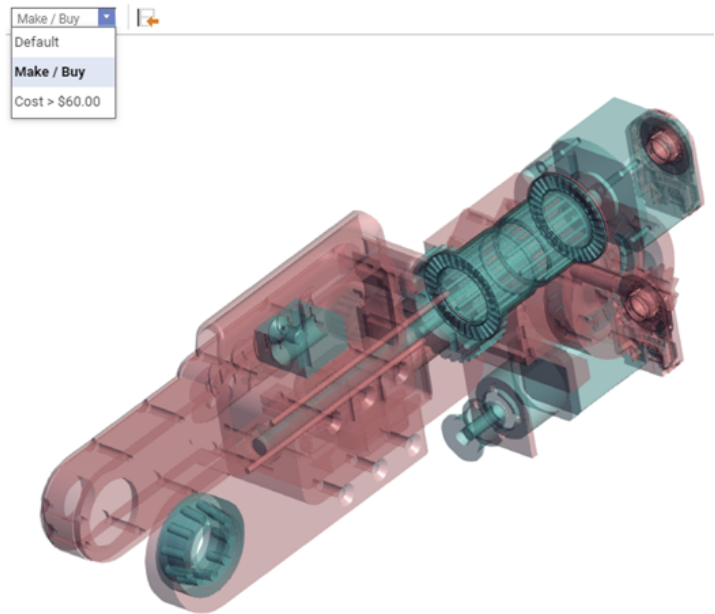
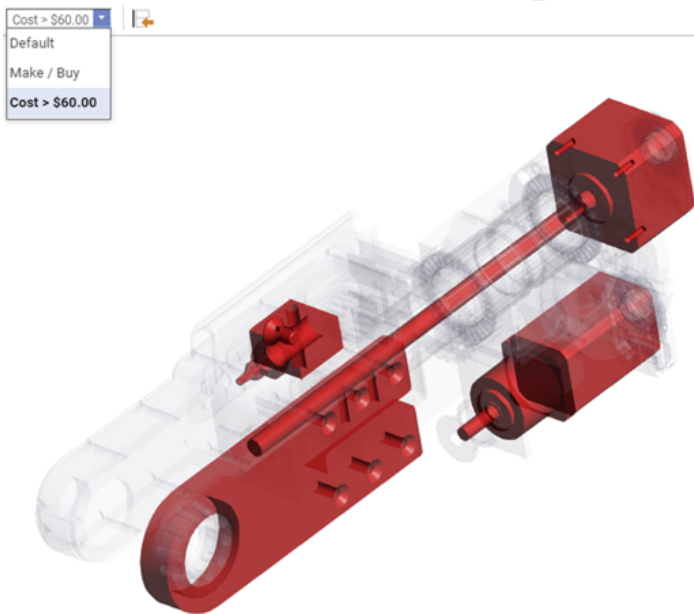
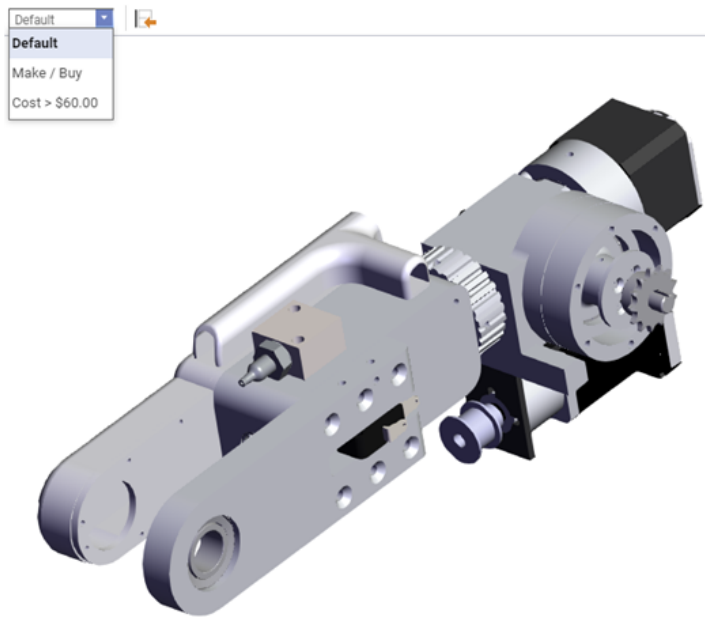
Processing combined rows in a custom query processor, such as when synchronizing CAD instances (described in section **Creating a list of Product Occurrences**) is possible by assigning the required properties (Query Reference Id and Item Id) for each combined row item to the relevant Product Occurrence instance.

For example, when combining the CAD Instance, CAD Structure, and CAD rows, the Items for all three rows will need to have `SetServiceProperty()` called to be able to synchronize the combined rows between the TGV and the Viewer.



## Rendering Configurations

Rendering Configurations are used to define alternate rendering properties (color, transparency) to apply to 3D geometry in the Dynamic or Streaming Viewer. The intent is to isolate/highlight certain 3D components or represent some aspect of related Items using color to differentiate from other 3D components. Each of these Rendering Configurations are accessible in the Dynamic or Streaming Viewer via the View Modes dropdown on the viewer toolbar.



The above figure shows sample View Modes created by the inclusion of Rendering Configurations (see code related to #5 in Section Processing Query Results). The top diagram shows the 3D geometry rendered using the default colors as defined when the CAD data was imported. The diagram on the lower left shows a View Mode that applies alternate colors to all 3D Components associated with Parts that have a cost greater than \$60. The diagram on the lower right shows a View Mode that renders parts in Coral that are manufactured and light Green that are purchased. Note the use of transparency (Opacity) in each example.

Each of the bottom two examples were created using two Rendering Configurations with a 'Name' that is displayed as a View Mode in the 3D Dynamic or Streaming Viewer. When creating Rendering Configurations, the name is what distinguishes them. There can be multiple Rendering Configurations added to the same set of Product Occurrences; each must have a unique Name.

The following code snippet shows an example of creating the Rendering Configuration shown in the bottom right above:

```
ProductOccurrenceRenderingConfiguration rConfigMakeBuy = new
ProductOccurrenceRenderingConfiguration();
rConfigMakeBuy.SetColor(c.isManufactured() ? Color.LightCoral :
Color.LightSeaGreen);
rConfigMakeBuy.Opacity = .5;
rConfigMakeBuy.Name = "Make / Buy";
poInst.RenderingConfigurations.Add(rConfigMakeBuy);
```

Rendering Configurations are defined using the

`ProductOccurrenceRenderingConfiguration` class and are added to each Product Occurrence Instance.

#### Important



It is required that all Product Occurrence Instance objects have a Rendering Configuration object added if View Modes are going to be used within a Query Processor.

Each unique View Mode must use the same string name value for the Name Property. Thus, a View Mode represents the collection of all Rendering Configuration Objects with the associated Name.

1. A Rendering Configuration Object is created.
2. Logic included in the Data Model Object determines whether the associated CAD Item is related to a manufactured Part. Note that when constructing Query Definitions for a Query Processor, additional Properties may be needed to be used solely for Rendering Configurations (View Modes).

The default color is black, and the default opacity is 0 (completely hidden). As mentioned, it is important to add a Rendering Configuration to each Instance to prevent default settings from applying.

#### **Important**

Use of 0 transparency does not prevent parts from being selected in the viewer. If an Opacity property is set to 0, the geometry will still be loaded, and it will be selectable even though the geometry isn't shown.

3. A unique Name is applied. It will be same for all Product Occurrences returned for the View Data.
4. The Rendering Configuration object is added to the Product Occurrence Instance object.



## Deploying a Query Processor

If the implementation of Query Processing uses the approach in this section – use of a separate DLL – the following steps can be used to compile the DLL and integrate it with the Aras Innovator Server.

### Important

The full implementation of the Custom Default Query Processor referenced in this Section is available in Section Sample Custom Default Query Processor.

The following steps outline the process to deploy a query processor:

1. Implement the DLL as defined in previous sections.
2. Follow the Steps in Section *Error! Reference source not found.* to install the Dynamic or Streaming Viewer.
3. Build and deploy the DLL.
4. Define the Data Processor Method.
5. Build Query Definition(s), Tree Grid View Definition(s), and Dynamic View Definition(s) and link with the Data Processor Method created.



## Build and deploy the DLL

The example described in this section was implemented using Microsoft Visual Studio with an output type of *Class Library* and a target Framework using *.Net Standard 2.0*.

### Important

DLLs integrated with Aras Innovator need to be signed.

The DLL must also be compiled by linking with the `Aras.DynamicModelViewer.DataModel` and `Aras.Server.Core` libraries which are included with the Dynamic Model View Install and with the standard Aras Innovator installation respectively. Note also that the Rendering Configuration classes use the `Color` class as defined in `System.Drawing`. The resulting DLL needs to be stored within the `< Aras Innovator Install Dir>/server/bin` directory. Once the DLL is stored, update the `method-config.xml` file in the `<Aras Innovator Install Dir>/server` directory to include a reference to the DLL. See the following figure for an example.

```
<MethodConfig>
  <ReferencedAssemblies>
    ...
    <name>$(binpath) /Aras.DynamicModelViewer.DataModel.dll</name>
    <name>$(binpath) /Aras.DynamicModelViewer.QueryProcessor.dll</name>
    <name>$(binpath) /CustomDefaultQP.dll</name>
    <name>$(binpath) /CustomPartQP.dll</name>
  </ReferencedAssemblies>
```

### Important

Due to the differences between Windows and Linux file systems it is required to use OS-specific path separator in paths. Remember that the Linux file system is case sensitive so there is significant difference between file names `./path/to/file.xml` and `./path/to/File.xml`. For more information about cross-platform development, see section 2.3 Cross-platform development in the Aras Innovator 31 - Programmer's Guide.



### Define the Data Processor Method

Create the Data Processor Method as described in the beginning of section *Create the Query Processor DLL*. Make sure that the appropriate level of validation is included. Note also that Query Parameters can be used to *parameterize* the custom Query Processor. Query Parameters are defined within the Query Definition and enabled in the Tree Grid View Definition. Parameters used within a Query Definition can be applied to the Where Conditions within the Query Definition, used for a Query Processor, or both. See Section *Query Parameters* for a description of creating a Query Parameter. Query Parameters are accessible using the `EventArgs` variable in the Data Processor Method.



### Create the Query/Tree Grid View/Dynamic View Definitions

Define the Query and View Definitions as described in this section. Note that related Item data can be added to the Query Definition and displayed in the Tree Grid View without being processed by the Query Processor. Doing so uses the 3D View as a navigation aid to access related Item data. When adding ItemTypes to a Query Definition, ensure to add the **ID** Property so the View context menu will be enabled by default. Likewise, related Item data can be added to a Query Definition to be processed solely by the Query Processor. Doing so can provide additional information when/if Rendering Configurations are included.



## Help files

The Installation files for the Dynamic and Streaming Viewer contain a directory `/QueryProcessorAPI` with API help files describing the classes/methods of the API.



## DynamicView Client-Side API

The `DynamicView` base class is a JavaScript (JS) object to execute functions associated with the Dynamic and Streaming Viewer, such as **Refresh**, **Isolate**, **Hide All**, etc. Its instances are accessible via JS Methods invoked by CUI Action Items for the Dynamic and Streaming Viewer Tree Grid View, Toolbar, and 3D View Canvas.

Developers can use the `DynamicView` client-side API to customize context menu commands and toolbar buttons to execute functionality that uses or processes 3D View or 3D Model data. The subsections of this section describe the `DynamicView` functions. Section *Customization Examples with DynamicView API* provides customization examples.



## Accessing DynamicView

To access the `DynamicView` base class, use the following code snippet:

```
// MethodTemplateName=CUI;  
const dynamicView = options.dpnContext.getDynamicView();
```

### `dynamicView.addModel(itemIds)`

The `dynamicView.addModel(itemIds)` function adds a model to a Dynamic or Streaming 3D Viewer scene as a separate geometry in 3DV and tree node on Tree Grid View.

The function has the following parameter:

`itemIds` : an array of respective item IDs that contain 3D geometry.

## `dynamicView.removeModel()`

The `dynamicView.removeModel()` function removes a model from a Dynamic or Streaming 3D Viewer scene and Tree Grid View.

## `dynamicView.isolateSelectedNodes()`

The `dynamicView.isolateSelectedNodes()` function does the following:

- Fits selected nodes into a Dynamic or Streaming 3D Viewer scene.
- Makes unselected nodes transparent.

## `dynamicView.setVisibilitySelectedNodes(isVisible)`

Depending on a passed parameter value, the

`dynamicView.setVisibilitySelectedNodes(isVisible)` function toggles the following:

- The visibility of selected nodes in a Dynamic or Streaming 3D Viewer scene.
- The visibility state of selected nodes in the Tree Grid View.

The function has the following parameter:

- `isVisible`: a Boolean flag
  - `true` : to show selected nodes in a Dynamic or Streaming 3D Viewer scene and set their state to `visible` in the Tree Grid View.
  - `false` : to hide selected nodes in a Dynamic or Streaming 3D Viewer scene and set their state to `hidden` in the Tree Grid View.

## `dynamicView.hideAllOtherNodes()`

The `dynamicView.hideAllOtherNodes()` function does the following:

- Fits selected nodes into a Dynamic or Streaming 3D Viewer scene.
- Hides unselected nodes in the Dynamic or Streaming 3D Viewer scene.
- Sets the visibility state of unselected nodes to `hidden` in the Tree Grid View.

## `dynamicView.fitAllNodes()`

The `dynamicView.fitAllNodes()` function fits all nodes into a Dynamic or Streaming 3D Viewer scene.

## `dynamicView.resetView()`

The `dynamicView.resetView()` function resets a model and Dynamic or Streaming 3D Viewer scene to their original states:

- Sets the camera to its default position.
- Makes all nodes visible in the Dynamic or Streaming 3D Viewer scene.
- Sets the visibility state of all nodes to visible in the Tree Grid View.

## `dynamicView.displayAllNodes()`

The `dynamicView.displayAllNodes()` function does the following:

- Makes all nodes visible in the Dynamic or Streaming 3D Viewer scene.
- Sets the visibility state of all nodes to visible in the Tree Grid View.



### `dynamicView.updateContextMenu()`

The `dynamicView.updateContextMenu()` function updates the visibility of context menu elements according to the nodes currently selected in the Tree Grid View.



## Using JT/STEP Converter

This section describes the installation steps of an optional JT/STEP Converter. This installation requires a System Administrator to make changes on the server where Aras Conversion Server is installed.

### Important

There is no Out of the Box Conversion Rule that will implement the JT/STEP Converter. In order to make use of this capability, custom **Conversion Rules** will need to be created. Please refer to Aras Innovator 31 - Example for Developing Conversion Server Tasks for guidelines.

The following steps outline the process of installing the JT/STEP Converter:

1. Copy and unzip Aras 3D Visualization 35 CD Image package on the server where Conversion Server is installed.
2. Go to Aras 3D Visualization 35 CD Image\Packages directory and copy the HOOPS Exchange folder.
3. Paste the copied folder to a permanent location on the server. Example: C:/HOOPS Exchange
4. Open **ConversionServerConfig.xml** file for edit and add the following tags:
  1. In the child sectionGroup name="ConverterSettings" tag of the configSections tag, add the following section tags with attributes:

```
<configSections>  
  
<sectionGroup name="ConverterSettings">  
  
...  
  
<section name="DpnCadConverterStepJt"  
type="Aras.ConversionFramework.Converter.Hoops.Configuration.HoopsConver  
ArasCadConverter" />  
  
</sectionGroup>  
  
</configSections>
```



2. In the child **Converters** tag of the **ConversionServer** tag, add the following **Converter** tags with attributes:

```
<ConversionServer>  
  
<Converters>  
  
...  
  
<Converter name="JT Step CAD Converter"  
type="Aras.CadConverter.StepJtCadConverter, ArasCadConverter" />  
  
</Converters>  
  
</ConversionServer>
```

3. In the **ConverterSettings** tag, add the following tags with child tags and attributes:

```
<ConverterSettings>  
  
...  
  
<DpnCadConverterStepJt>  
  
<Application converterPath="C:\H00PS Exchange\bin\win64_v142" />  
  
</DpnCadConverterStepJt>  
  
</ConverterSettings>
```

5. Save and close **ConversionServerConfig.xml** file.

## Appendix

### Sample Custom Default Query Processor

This section includes the complete sample code discussed in section *Create the Query Processor DLL*. The following class represents the main class used for custom Query Processing. It contains the `processQueryResults` method discussed in section *Create the Data Processor Method* and the build method discussed in section *Creating a list of Product Occurrences*.

```
using System;

using System.Collections.Generic;

using Aras.DynamicModelViewer.DataModel; // Required for Product
Occurrences

using Aras.Server.Core.QueryBuilder; // Required to Process Query
results

using System.Drawing; // Required for Colors used in Rendering
Configurations

namespace CustomDefaultQP
{
    /// <summary>
    /// Main Class for processing the results from the execution of the
    default
    /// Query Definition used for Dynamic Visualization
    /// </summary>
```



```
public class DefaultQueryResultsParser
{
    /// <summary>
    /// Default Constructor
    /// </summary>
    public DefaultQueryResultsParser() { }

    /// <summary>
    /// Processes the given Query Result Items to build a hierarch of CAD
    Items
    /// </summary>
    /// <param name="resultItems">Input set of top-level Query
    Results</param>
    /// <returns>List of Product Occurrences</returns>
    public ICollection<ProductOccurrenceBase>
    processQueryResults(IEnumerable<QueryBuilderNode> resultItems)
    {
        // Process each root CAD Item in the query results
        List<CAD> rootCADItems = new List<CAD>();
        foreach (var resultItem in resultItems)
        {
```



```
if (resultItem.QueryItem.Alias == "CAD")
{
CAD nextCAD = new CAD();
nextCAD.processCAD(resultItem); // recurses through full hierarchy
rootCADItems.Add(nextCAD);
} // if()
} // foreach()

// Build the Product Occurrences for each root CAD Item
ICollection<ProductOccurrenceBase> poList = new
List<ProductOccurrenceBase>();
foreach (CAD c in rootCADItems)
{
ProductOccurrenceInstance poInst = new ProductOccurrenceInstance();
poInst.Attributes.Add(new ProductOccurrenceAttr("QUERY ITEM REF ID",
c.QryRefId));
poInst.Attributes.Add(new ProductOccurrenceAttr("ITEM ID", c.ID));

poInst.Name = c.Name;

if (c.Children.Count == 0 && c.SCFileID != null) // Assume this is a
component part
```



```
{  
    poInst.ProductOccurrenceSource.Name = c.SCFileID; // use File Item Id for  
    name  
    poInst.ProductOccurrenceSource.FileReferenceByTypeMap.Add(SourceModelType.Scs,  
    c.SCFileID);  
}  
  
if (c.Children.Count == 0 && c.SCFileID == null)  
    poInst = null; // Invalid - no view file attached  
else  
{  
    poInst.SetAsRoot(); // root assembly,  
    poInst.SetServiceProperty(c.QryRefId, c.ID); // add service properties  
    poList.Add(poInst); // add to the PO list  
}  
  
// Recurse through child hierarchy  
foreach (CAD child in c.Children)  
    build(poInst, child);  
} // foreach (CAD c in rootCADItems)  
return poList;
```



```
} // processQueryResults()  
  
/// <summary>  
/// Constructs a list of Product Occurrences (PO) from the CAD Items  
parsed  
/// from the Query Results. This will construct a unique Product  
Occurrence  
/// for each instance of a part.  
/// </summary>  
/// <param name="parent"> Product Occurrence to attach newly created POs  
to.  
/// This value can't be null</param>  
/// <param name="c">CAD Item representing the child CAD data to build  
from</param>  
private void build(ProductOccurrenceInstance parent, CAD c)  
{  
// Create a Product Occurrence for each instance of the given CAD  
foreach (CADInstanceInfo cadInstInfo in c.Instances)  
{  
ProductOccurrenceInstance poInst = new ProductOccurrenceInstance();
```



```
poInst.Attributes.Add(new ProductOccurrenceAttr("QUERY ITEM REF ID",  
c.QryRefId));  
  
poInst.Attributes.Add(new ProductOccurrenceAttr("ITEM ID", c.ID));  
  
// bounding box  
  
poInst.ProductOccurrenceSource.BoundingBox.Max.X = c.BBox.MaxX;  
poInst.ProductOccurrenceSource.BoundingBox.Max.Y = c.BBox.MaxY;  
poInst.ProductOccurrenceSource.BoundingBox.Max.Z = c.BBox.MaxZ;  
poInst.ProductOccurrenceSource.BoundingBox.Min.X = c.BBox.MinX;  
poInst.ProductOccurrenceSource.BoundingBox.Min.Y = c.BBox.MinY;  
poInst.ProductOccurrenceSource.BoundingBox.Min.Z = c.BBox.MinZ;  
  
// *** Testing Rendering Configuration ***  
  
ProductOccurrenceRenderingConfiguration rConfig = new  
ProductOccurrenceRenderingConfiguration();  
  
rConfig.SetColor(Color.FromArgb(55, 40, 0));  
  
rConfig.Opacity = 0.4;  
  
rConfig.Name = "test";  
  
poInst.RenderingConfigurations.Add(rConfig);  
  
// *** Testing Rendering Configuration ***
```



```
// **** ADD SetServiceProperty METHOD CALLS ****  
  
poInst.TransformationMatrix = cadInstInfo.XForm;  
  
poInst.SetServiceProperty(cadInstInfo.RefID, cadInstInfo.ID);  
  
poInst.SetServiceProperty(c.CADStrQryRefId, c.CADStructureID);  
  
poInst.SetServiceProperty(c.QryRefId, c.ID);  
  
// **** ADD SetServiceProperty METHOD CALLS ****  
  
  
poInst.Name = c.Name;  
  
if (c.Children.Count == 0 && c.SCFileID != null) // Assume this is a  
component part  
{  
  
    poInst.ProductOccurrenceSource.Name = c.SCFileID; // use File Item Id for  
name  
  
    poInst.ProductOccurrenceSource.FileReferenceByTypeMap.Add(SourceModelType.Sc, c.SCFileID);  
  
}  
  
if (c.Children.Count == 0 && c.SCFileID == null)  
poInst = null; // Invalid - no view file attached  
else  
parent.Children.Add(poInst);
```



```
// Recurse through child hierarchy
foreach (CAD child in c.Children)
    build(poInst, child);
} // foreach(String xFormStr in c.Instances)

} // build()

} // class DefaultQueryResultsParser
} // namespace CustomDefaultQP
```

The following `CAD` class is used to collect information from the Query Results. It contains the `processCAD` method discussed in section *Processing Query Results*.

```
using System;
using System.Collections.Generic;
using Aras.Server.Core.QueryBuilder;

namespace CustomDefaultQP
{
    struct BoundingBox
    {
        public double MinX; // X Coord for minimum point
```



```
public double MaxX; // X Coord for maximum point
public double MinY; // Y Coord for minimum point
public double MaxY; // Y Coord for maximum point
public double MinZ; // Z Coord for minimum point
public double MaxZ; // Z Coord for maximum point
} // class BoundingBox

/// <summary>
/// Stores information for each CAD Instance Query Item
/// </summary>
class CADInstanceInfo
{
public CADInstanceInfo()
{
XForm = "1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1"; // Default Identity Matrix
ID = "0";
}

/// <summary>
/// CAD Instance transformation matrix
```



```
/// </summary>
internal String XForm { get; set; }
}
/// <summary>
/// CAD Instance Id
/// </summary>
internal String ID { get; set; }
}
/// <summary>
/// CAD Query Reference Id
/// </summary>
internal String RefID { get; set; }
} // class CADInstanceInfo
}
/// <summary>
/// Stores all data necessary for creating 3D View data assuming
/// it is contained within 'CAD' Query Items
/// </summary>
```



```
class CAD
{
private BoundingBox _BBox; // Bounding volume

/// <summary>
/// Default Constructor
/// </summary>
internal CAD()
{
// Ensure that at least one instance is added
// This will be replaced in _processCADStructure() if instances
// are included in the query results
Instances.Add(new CADInstanceInfo());
} // CAD()

/// <summary>
/// Processes a given CAD Item contained within the given
<code>QueryBuilderNode</code>
/// </summary>
/// <param name="qbItem">Item containing properties for CAD
Items</param>
```



```
/// <exception cref="ArgumentException">Required Properties not included
in results</exception>
internal void processCAD(QueryBuilderNode qbItem)
{
    QryRefId = qbItem.QueryItem.RefId;
    try
    {
        ID = qbItem.GetProperty("id"); // required
        Name = qbItem.GetProperty("name"); // required
    }
    catch
    {
        throw new ArgumentException("Query Definition is not defined properly:
'id' and 'name' Properties are required for CAD Items");
    }

    // Bounding Box Data. Alternate values ensure that there is a non-empty
    // bounding volume defined
    // NOTE: Bounding box data should first be retrieved from the
    // related CAD ConversionInfo Relationship. If not present,
    // retrieve from the CAD Item directly
```



```
_BBox.MinX = _getPropertyAsDouble(qbItem, "x_min", -1.0);  
_BBox.MaxX = _getPropertyAsDouble(qbItem, "x_max", 1.0);  
_BBox.MinY = _getPropertyAsDouble(qbItem, "y_min", -1.0);  
_BBox.MaxY = _getPropertyAsDouble(qbItem, "y_max", 1.0);  
_BBox.MinZ = _getPropertyAsDouble(qbItem, "z_min", -1.0);  
_BBox.MaxZ = _getPropertyAsDouble(qbItem, "z_max", 1.0);  
  
// for processing CAD children and associated view file  
foreach (var child in qbItem.ChildNodes)  
{  
    if (child.QueryItem.Alias == "CAD ConversionInfo")  
        _processCADConversionInfo(child);  
    if (child.QueryItem.Alias == "CAD Structure")  
        _processCADStructure(child);  
    if (child.QueryItem.Alias == "File")  
        _extractFileInfo(child);  
} // foreach  
} // processCAD(QueryBuilderInterface qbItem)  
  
/// <summary>
```



```
/// Processes the bounding box data in the given CAD ConversionInfo node
/// </summary>
/// <param name="qbItem"></param>
private void _processCADConversionInfo(QueryBuilderNode qbItem)
{
    _BBox.MinX = _getPropertyAsDouble(qbItem, "x_min", -1.0);
    _BBox.MaxX = _getPropertyAsDouble(qbItem, "x_max", 1.0);
    _BBox.MinY = _getPropertyAsDouble(qbItem, "y_min", -1.0);
    _BBox.MaxY = _getPropertyAsDouble(qbItem, "y_max", 1.0);
    _BBox.MinZ = _getPropertyAsDouble(qbItem, "z_min", -1.0);
    _BBox.MaxZ = _getPropertyAsDouble(qbItem, "z_max", 1.0);
} // _processCADConversionInfo(QueryBuilderNode qbItem)

/// <summary>
/// Returns the value of the given Property as a double, use given
alternate
/// if the value is not set or is not defined.
/// </summary>
/// <param name="qbItem">Query Builder Item containing the
Property</param>
/// <param name="propName">Property Name</param>
```



```
/// <param name="alt">Used when given property not set</param>
/// <returns>Value if Property exists and is set, 'alt' value
otherwise</returns>

private double  getPropertyAsDouble(QueryBuilderNode qbItem, String
propName, double alt)
{
// check for existence of Property (in future)

if (qbItem.TryGetProperty(propName, out string tmpPropVal) && tmpPropVal
!= null)
return Double.Parse(tmpPropVal);
else
return alt;
} // getPropertyAsDouble()

/// <summary>
/// Processes the child CAD Instances and Files associated with the
given
/// <code>QueryBuilderItem</code>
/// </summary>
/// <param name="qbItem"></param>
/// <exception cref="ArgumentException">Required Properties not included
in results</exception>
```



```
private void _processCADStructure(QueryBuilderNode qbItem)
{
    // Save the instances and store with child CAD Item
    List<CADInstanceInfo > curInstances = new List<CADInstanceInfo>(); CAD
nextCAD = new CAD();
    nextCAD.CADStructureID = qbItem.ItemId;
    nextCAD.CADStrQryRefId = qbItem.QueryItem.RefId;
    // Loop through all CAD Instances and CAD Items
    foreach (var child in qbItem.ChildNodes)
    {
        try
        {
            if (child.QueryItem.Alias == "CAD Instance" &&
child.TryGetProperty("transformation_matrix", out string tmpXForm) &&
tmpXForm != null) {
                CADInstanceInfo cadInfo = new CADInstanceInfo();
                cadInfo.XForm = tmpXForm;
                cadInfo.ID = child.ItemId;
```



```
cadInfo.RefID = child.QueryItem.RefId;
curInstances.Add(cadInfo);
}
if (child.QueryItem.Alias == "CAD")
{
nextCAD.processCAD(child);
Children.Add(nextCAD);
}
} // try
catch { }
} // foreach()
if (curInstances.Count > 0)
nextCAD.Instances = curInstances;
} // _processCADStructure(QueryBuilderNode qbItem)

/// <summary>
/// Returns the first found <code>QueryBuilderNode</code> with the
/// given alias that is a descendant of the given Item. Note that if
/// the given Item has the alias, it is returned
/// </summary>
```



```
/// <param name="qbItem">Item containing descendants to search</param>
/// <param name="alias">Alias Name of Item to search for</param>
/// <returns>NULL, if not found; first found node with given alias
otherwise</returns>

private QueryBuilderNode _findChildWithAlias(QueryBuilderNode qbItem,
String alias)
{
if (qbItem.QueryItem.Alias == alias)
return (qbItem);
else
{
foreach (var child in qbItem.ChildNodes)
{
QueryBuilderNode descItem = _findChildWithAlias(child, alias);
if (descItem != null)
return (descItem);
} // for()
} // else
return (null);
} // findChildWithAlias(QueryBuilderNode qbItem, String alias)
```



```
/// <summary>
/// Extracts the SCS file data from the given
<code>QueryBuilderNode</code>. This
/// method assumes that the Properties for 'id' and 'filename' exist
/// in the given results node with alias 'File_1'
/// </summary>
/// <param name="qbItem"></param>
/// <exception cref="ArqumentException">Required File Properties not
included in results</exception>
private void _extractFileInfo(QueryBuilderNode qbItem)
{
QueryBuilderNode child = _findChildWithAlias(qbItem, "File_1");
if (child != null)
{
// filename and id are required for Product Occurrences
if (child.TryGetProperty("filename", out string tmpFileName) &&
tmpFileName != null &&
child.TryGetProperty("id", out string tmpId) &&
tmpId != null)
{
SCFile = tmpFileName;
}
```



```
SCFileID = tmpId;
}
else
throw new ArgumentException("Query Definition is not defined properly:
'filename' and 'id' Properties are required for view files");
} // if (child != null)
} // _extractFileInfo(QueryBuilderItem qbItem)

/// <summary>
/// Returns the name of the CAD Query Response Item
/// </summary>
internal String Name { get; private set; }

/// <summary>
/// Returns the Query Reference ID from the CAD Query Response Item
/// used to create this CAD Object
/// </summary>
internal String QryRefId { get; private set; }
```



```
/// <summary>  
/// Returns the Query Reference ID from the CAD Structure Query  
/// Response Item used to create this CAD Object  
/// </summary>
```

```
internal String CADStrQryRefId { get; private set; }
```

```
/// <summary>  
/// Returns the Id of the CAD Query Response Item  
/// </summary>
```

```
internal String ID { get; private set; }
```

```
/// <summary>  
/// Returns the Id of the CAD Query Response Item  
/// </summary>
```

```
internal String CADStructureID { get; private set; }
```

```
/// <summary>
```



```
/// Returns the SCS/SCZ File name associated with the CAD Query Response  
Item  
/// </summary>  
internal String SCFile { get; private set; }  
  
/// <summary>  
/// Returns the SCS/SCZ File ID associated with the CAD Query Response  
Item  
/// </summary>  
internal String SCFileID { get; private set; }  
  
/// <summary>  
/// Returns the list of data for CAD Instance Query Response Items  
/// </summary>  
internal List<CADInstanceInfo> Instances { get; private set;} = new  
List<CADInstanceInfo>();  
  
/// <summary>  
/// List of child CAD Items as determined by the CAD Structure  
/// </summary>  
internal List<CAD> Children { get; } = new List<CAD>();
```



```
/// <summary>  
/// Bounding Volume  
/// </summary>  
internal BoundingBox BBox { get { return _BBox; } }  
  
} // class CAD  
} // namespace CustomDefaultQP
```



## Customization Example with DynamicView API

With the DynamicView client-side API, developers can customize the Dynamic or Streaming Viewer context menu commands and toolbar buttons. This section provides an example of such customization to give a solid understanding of the customization process. For the DynamicView client-side API description, see section DynamicView Client-Side API.

The customization process has two stages:

1. Writing custom JS functions for the required functionality.
2. Creating context menu commands using the standard Aras Innovator capabilities.

The example shows how to create the custom **Add Item(s) To Change** TGV Context Menu item that calls the **Choose Change Item** dialog for selected nodes.

## Writing a Custom JS Function

1. In Innovator/Client/Solutions/3DV/Scripts/Controls/TGV/DynamicView.ts add the following function:

```
this.getSelectedItemsHandler = function () {  
    var ccItem;  
    const tgvGrid = mainPageExtension.getTgvGrid();  
    const treeGrid = mainPageExtension.getTreeGrid();  
    const selectedRowsIds = treeGrid.settings.selectedRows;  
    const count = selectedRowsIds ? selectedRowsIds.length : 0;  
  
    // Add a dummy item to start the result collection (removed later)  
    var resultItem = aras.newIOMItem("DeleteMe", "DeleteMe");  
    for (let i = 0; i < count; i++) {  
        const rowId = selectedRowsIds[i];  
        const selectedRow = tgvGrid.getRow(rowId);  
        const dataStr = selectedRow.cells[0].data;  
        if (dataStr) {  
            const data = JSON.parse(dataStr);  
            var thisType = data.type; // this.getType();  
            var thisId = data.id;  
            var relshipItemType;
```



```
var sourceItemTypeName;  
  
var isRelationship = false;  
  
// Check whether the method is being called from a relationship grid, the  
main grid or on an  
  
// individual item and build an array of item ids  
  
var relTypeId = aras.getRelationshipTypeId(thisType);  
  
if (relTypeId) {  
    isRelationship = true;  
  
    relshipItemType = aras.getRelationshipType(relTypeId);  
  
    var sourceItemType =  
aras.getItemTypeDictionary(relshipItemType.getProperty("source_id"),  
"id");  
  
    sourceItemTypeName = sourceItemType.getProperty("name");  
}  
  
// When starting from a relationship, get the parent item from the cache  
and then retrieve the related item  
  
if (isRelationship) {  
  
    var parentId = relshipItemType.getProperty("source_id", "");
```



```
var parentItem = aras.getItemById(sourceItemTypeName, parentId, 0);  
if (parentItem) {  
    ccItem = parentItem.selectSingleNode("Relationships/Item[@id='" + thisId  
+ "']");  
  
    //If item isn't cached, get  
    if (!ccItem) {  
        ccItem = aras.getItemById(thisType, thisId, 0);  
    }  
  
    //get related Part  
    if (ccItem) {  
        ccItem = aras.getRelatedItem(ccItem);  
    }  
}  
  
else {  
    ccItem = aras.getItemById(thisType, thisId, 0);  
}
```



```
// Add the item to the collection to be returned

if (ccItem) {
    if (aras.isNew(ccItem)) {
        aras.AlertError(aras.getResource("PLM",
"affecteditem.add_not_saved_item", aras.getKeyedNameEx(ccItem)));
        return;
    }
    var tempItem = aras.newIOMItem("", "");
    tempItem.loadAML(ccItem.xml);
    resultItem.appendItem(tempItem);
}
}
}

if (resultItem.getItemCount() < 2) { return; }

// Remove the dummy item
resultItem.removeItem(resultItem.getItemByIndex(0));

// Set the ChangeItem attribute for debugging purposes and return the
results
```



```
resultItem.getItemByIndex(0).setAttribute("ChangeItem", "Pending");  
  
var methodArgs = {};  
methodArgs.results = resultItem;  
results = aras.evalItemMethod(  
  'PE_ChoseCMItem',  
  ccItem,  
  methodArgs  
);  
return resultItem;  
};
```

This function calls a new function which is an adapted version of the standard **PE\_GetSelectedItems** Server Method.

2. Create the **dpn\_GetSelectedItems** Method of the **JavaScript** Method Type with its execution allowed to the **World** Identity:

```
// MethodTemplateName=CUI;  
const dynamicView = options.dpnContext.getDynamicView();  
dynamicView.getSelectedItemsHandler ();
```

This Method will be a menu item click handler.



## Creating a New TGV Context Menu Item

The following steps outline the process to create a new TGV context menu item:

1. Go to Contents and select Administration.
2. Select Configuration.
3. Click Tree Grid Views and open the View3D\_CAD Item.

**View3D\_CAD** ☆ 🚩

**Edit** ↻ ↶ 🌳 🗑️ 📄 ⋮

Tree Grid View

**Name** View3D\_CAD **Query Definition** [View3D\\_CAD](#)

**Context Item Type** [CAD](#)

**Description**  
Default Tree Grid View Definition for Dynamic 3D viewer - DO NOT REMOVE

**Max Visible Children On Expand** 100 **Linked Toolbar/Context Menu** [View3D\\_CAD Presentation Configuration](#)

**Max Grow Levels** 2

**Auto Grow On Refresh**

4. Open the View3D\_CAD Presentation Configuration Item.



**View3D\_CAD Presentation Configuration** ☆

**Edit** [Refresh] [Undo] [Tools] [Trash] [Share] [More]

^ **Presentation Configuration**

Color [Input]  
Name  
View3D\_CAD Presentat

---

^ **Command Bar Section** cui\_PresentConfigWinSection Presentation Command

**CommandBarSection** ☆

[Icons] [Hidden] [Tools] [More]

Classification	Name	Bui...	Label	Description	Additional Da...	Location [...]	sort_order	For Identity [...]	For Classifica...
Data Model	com.aras.inno...		Dynamic 3D Viewer TGV Toolbar			<a href="#">TGV_Toolbar</a>	128	<a href="#">World</a>	
Data Model	com.aras.inno...		Dynamic 3D Viewer TGV Context Menu			<a href="#">TGV_ContextMenu</a>	256	<a href="#">World</a>	

< Prev Next > Page: 1 of 1 | 2 Results | [Filter]

5. Open the **Dynamic/Streaming 3D Viewer TGV Context Menu** Item from the **Command Bar Section** Relationships Grid.
6. Add a new **CommandBarItem** Relationship Item of the **Menu** ItemType.



The screenshot shows the Aras Innovator interface for configuring a **Command Bar Item**. The **Command Bar Section** has the following properties:

- Name: com.aras.innovator.viewers.dynamic\
- Location: TGV\_ContextMenu
- Classification: Data Model
- Label: Dynamic 3D Viewer TG\

The **Command Bar Item** configuration shows a table of items:

Item Type	Name	Additional Da...	Init Method [...]
Menu	dpn_TGVIso...		dpn_context_menu_item_...
Menu	dpn_TGVHide...		dpn_context_menu_item_...
Menu	dpn_TGVSho...		dpn_context_menu_item_...
Menu	dpn_TGVHide...		dpn_context_menu_item_...
Menu	dpn_TGVFitAll		
Menu	dpn_TGVRese...		
Menu	dpn_TGVDispl...	1280	Add




The **Select ItemType** dialog box is open, showing the following options:








- Button
- Checkbox
- DropDown
- Edit
- Menu** (Selected)
- Menu Button
- Menu Checkbox
- Menu Separator
- Separator
- Shortcut

7. Set the properties of the new **CommandBarItem** Item as follows:

1. Name: dpn\_GetSelectedItems
2. Label: Add Item(s) To Change
3. Init Method: dpn\_context\_menu\_item\_init
4. Click Method: dpn\_GetSelectedItems



 **dpn\_GetSelectedItems**  

 **Edit**   |   |  | 

**Menu**

**Name**  
dpn\_GetSelectedItems

**Label**                      **ARIA Label**  
Add Item(s) To Change     

**Tooltip Template**              **Command Alias**

**Parent Menu**

**Init Method**                      **Include Events**  
[dpn\\_context\\_menu\\_item\\_init](#)       ▲  
**Click Method**                       ■  
[dpn\\_GetSelectedItems](#)               ■  
    ▼

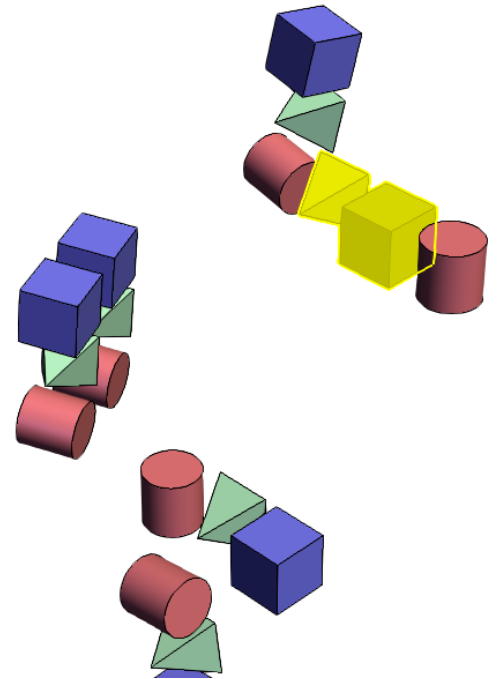
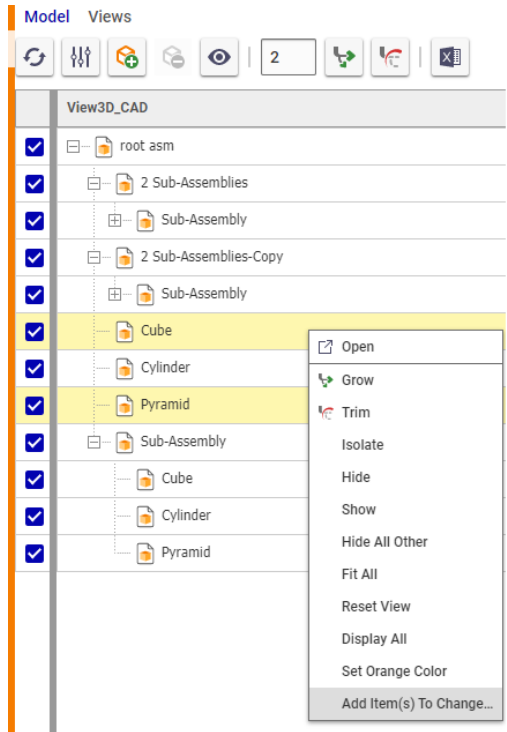
8. In the **Command Bar Section Relationships Grid**, set the properties of the new row as follows:

- Action: Add
- Identify: World



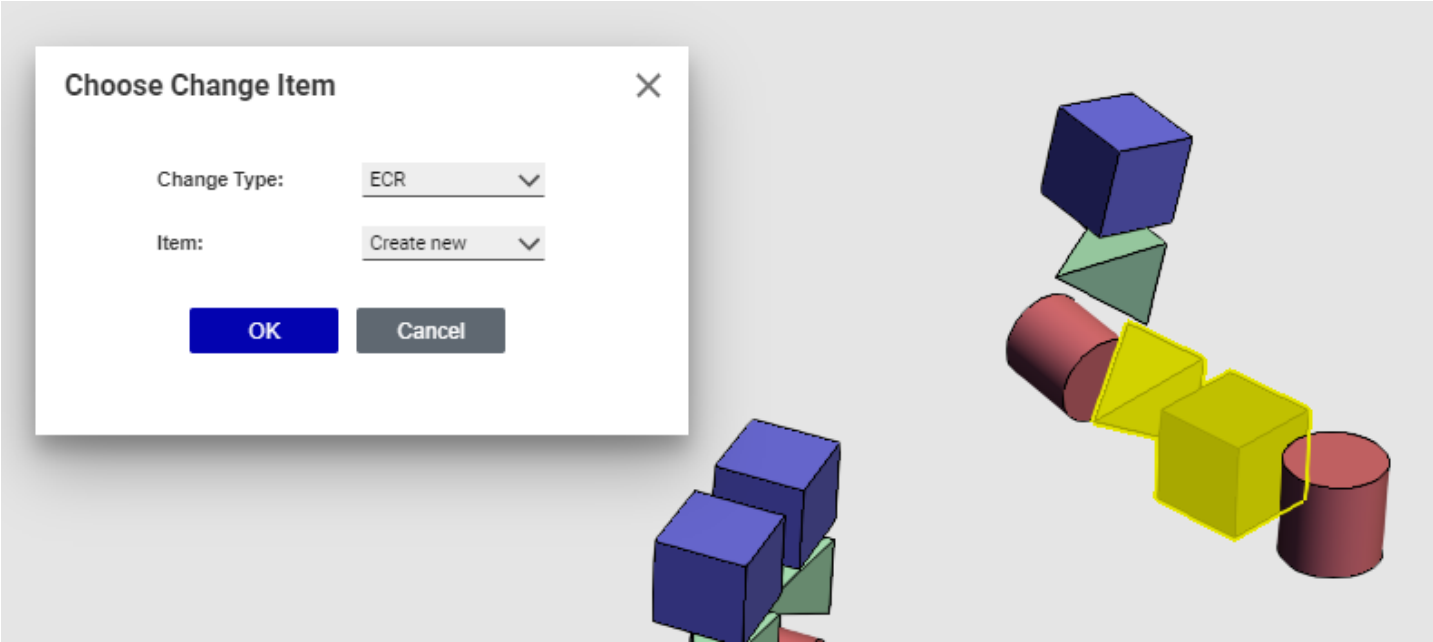
### TGV Context Menu Customization Results

The TGV Context Menu has the new **Add Item(s) To Change** command.



Clicking this command calls the **Choose Change Item** dialog for selected nodes.





## Reverse Proxy Server

This section is for information purposes only.

For Streaming Viewer, the Client-side 3D HOOPS Viewer will need to communicate with the Server-side Services using networking ports that are not typically open in customer firewalls. Reverse Proxies allow network traffic to use standard HTTP connections (80, 443) for such information flow.

The following steps outlines the process of installing and configuring the Reverse Proxy for Streaming 3D Viewer:

1. Install Application **Request Routing** (ARR) 3.0 from the following link: <https://www.iis.net/downloads/microsoft/application-request-routing> .

### Download ARR 3.0

---

- [x86 installer](#) / [x64 installer](#)

2. Install **URL Rewrite 2.1** from the following link:

<https://www.iis.net/downloads/microsoft/url-rewrite>

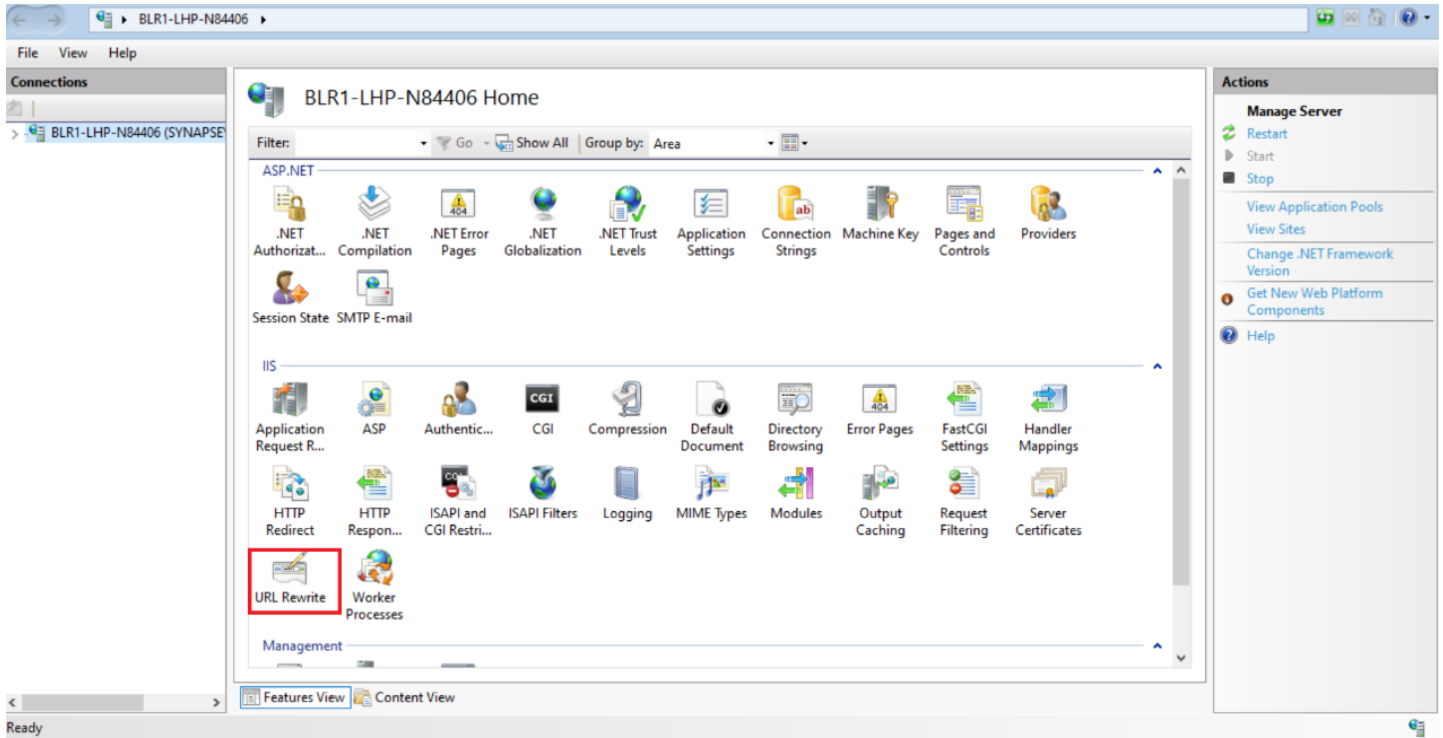
### Download URL Rewrite Module 2.1

---

- **English:** [x86 installer](#) / [x64 installer](#)

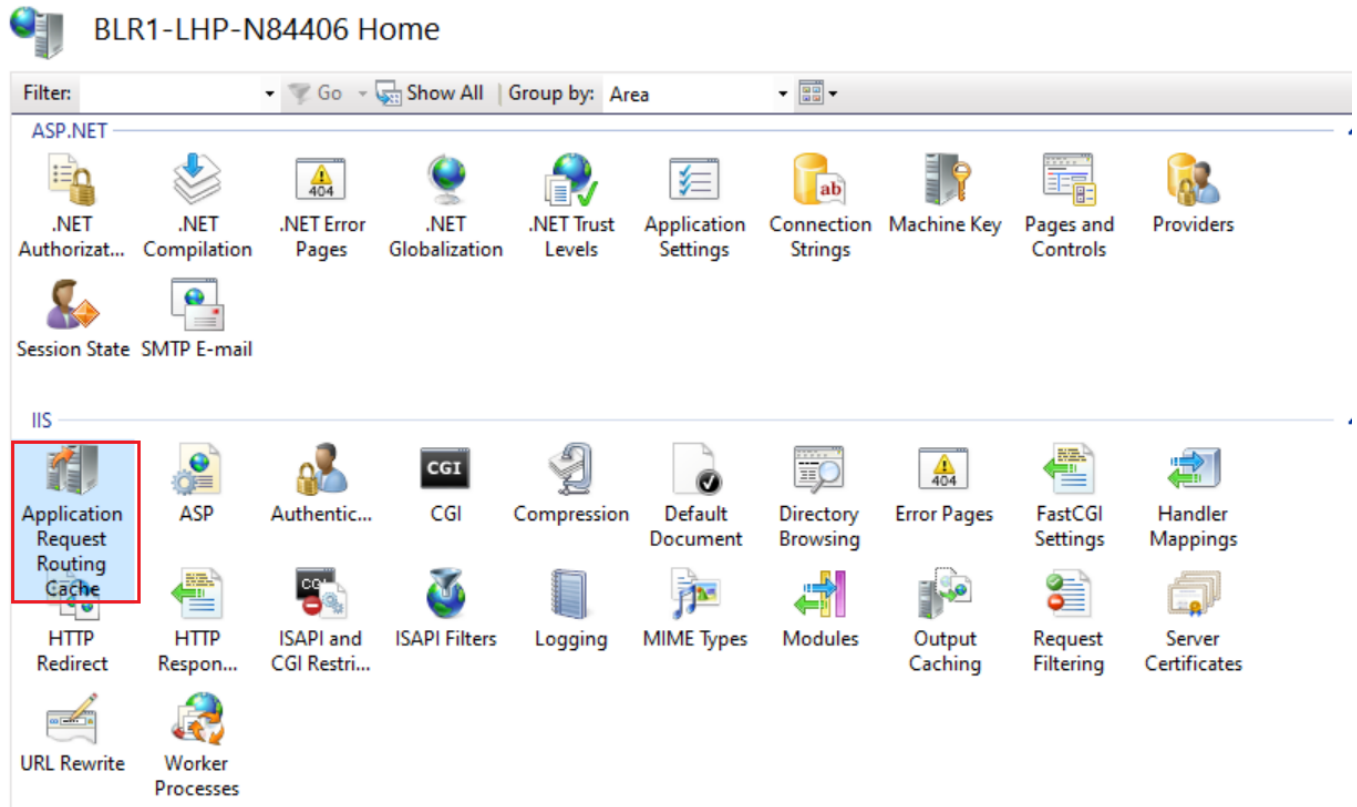
3. After installation is complete, open **IIS** from **Windows** browser. A new module appears in IIS Manager.



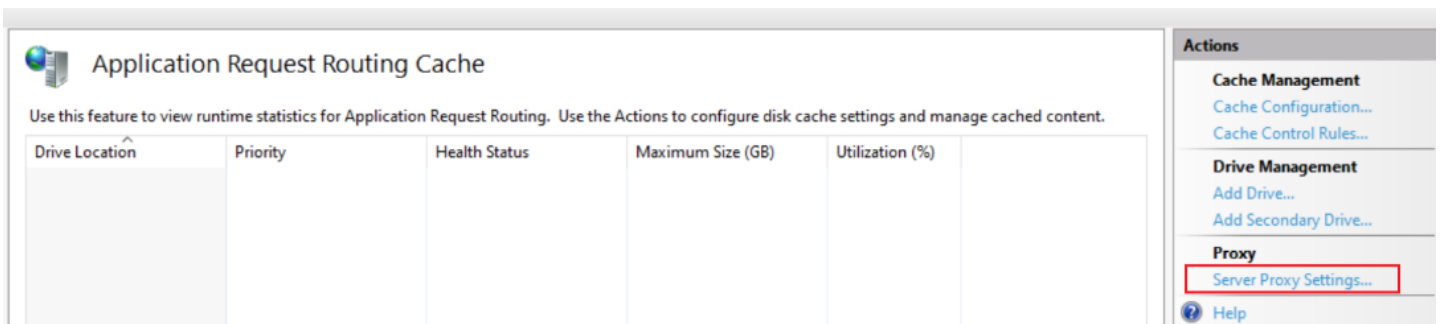


4. On IIS, click Application Request Routing Cache.





5. From the Actions column, click Server **Proxy Settings**.



6. Select **Enable Proxy** checkbox.





# Application Request Routing

Use this feature to configure proxy settings for Application Request Routing.

Enable proxy

Proxy Setting

HTTP version:

Pass through

Keep alive

7. Click Apply.

Next step is to Configure URL Rewrites rules. This can be done in two ways:

- Option 1: Configure URL Rewrite rules using **IIS**
- Option 2: Configure URL Rewrite rules using **web.config** file.

Option 1: Configure URL Rewrite rules using IIS:

The following two rules must be set up in URL Rewrite:

Entry point with link to Hoops Server

Web Socket connections

8. From Actions column, click **Add Rules**.

URL Rewrite

Provides rewriting capabilities based on rules for the requested URL address and the content of an HTTP response.  
Inbound rules that are applied to the requested URL address:

Name	Input	Match	Pattern	Action Type	A

Actions

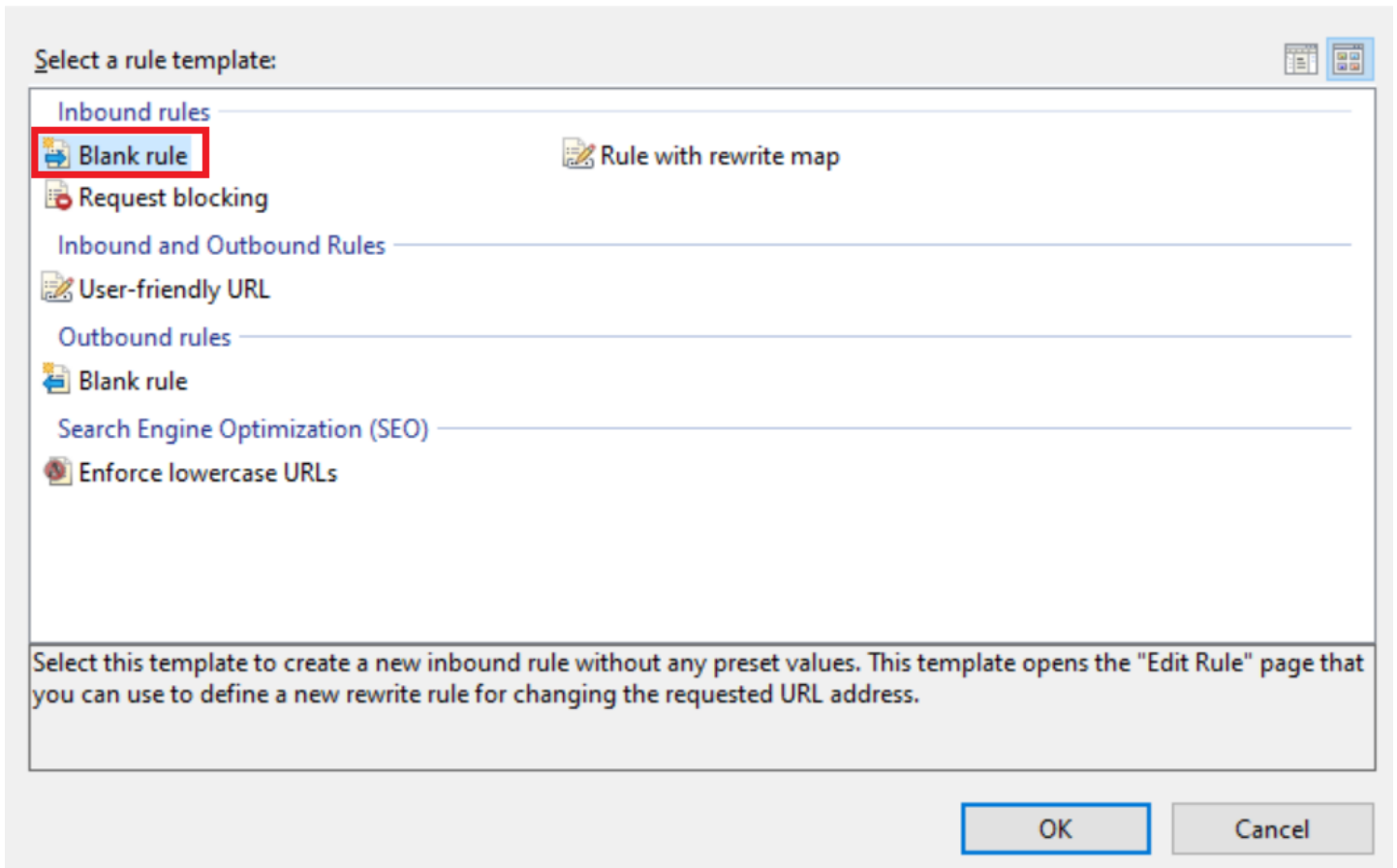
- Add Rule(s)...
- Manage Server Variables
  - View Server Variables...
- Manage Providers
  - View Rewrite Maps...
  - View Providers...

9. Click Blank Rule.



Add Rule(s)

? ×



10. Add the following details to the rule:

- Name: Hoops\_server
- Pattern: hoops\_server/?(.\*)
- Rewrite URL: <http://localhost:{R:1}>



## Edit Inbound Rule

Name:

Hoops\_server

### Match URL

Requested URL:

Matches the Pattern

Using:

Regular Expressions

Pattern:

hoops\_server/?(.\*)

Test pattern...

Ignore case

### Conditions

### Server Variables

### Action

Action type:

Rewrite

#### Action Properties

Rewrite URL:

http://localhost:{R:1}

Append query string

Stop processing of subsequent rules

11. Click **Apply**.

12. Add another Blank Rule with the following details:

- Name: Web Socket Reverse
- Pattern: ws(.\*)hoops\_server/?(.\*)



- Rewrite URL: ws://localhost:{R:2}

**Edit Inbound Rule**

**Name:**  
Web Socket Reverse

**Match URL**

**Requested URL:** Matches the Pattern

**Using:** Regular Expressions

**Pattern:** ws(?:hoops\_server/?).\* Test pattern...

Ignore case

**Conditions**

**Server Variables**

**Action**

**Action type:** Rewrite

**Action Properties**

**Rewrite URL:** ws://localhost:{R:2}

Append query string

Stop processing of subsequent rules

Option 2: Configure URL Rewrite rules using web.config file:

13. Open web config file from Aras Innovator code tree: "C:\Aras\InnovatorServer\Innovator\web.config file".
14. In the `<system.webserver>` tag, add the following code as shown in the screenshot below:



```

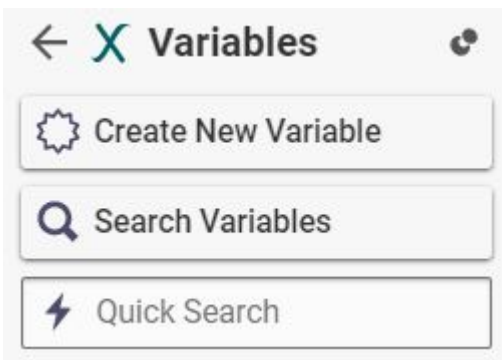
<system.webServer>
  <rewrite>
    <rules>
      <rule name="Web Socket Reverse" enabled="true" stopProcessing="true">
        <match url="ws(.*)hoops_server/?(.*)" />
        <action type="Rewrite" url="ws://localhost:{R:2}" />
      </rule>
      <rule name="hoops_server" enabled="true" stopProcessing="true">
        <match url="hoops_server/?(.*)" />
        <action type="Rewrite" url="http://localhost:{R:1}" />
      </rule>
    </rules>
  </rewrite>
</system.webServer>

```

- Hoops\_server in match url can be different and this will be accounted as "Hoops Server Uri".
- Action url should point to Instance of the Hoops Server.

Next step is to adjust **HOOPS\_ServerUrl**.

15. In the Aras Innovator Instance, from the **Table of Contents**, expand **Administration** and select **Variables**.
16. Click Search Variables.



17. Click **Search** and select **HOOPS\_ServerUrl**. The HOOPS\_ServerURL form appears.

X HOOPS\_ServerUrl ☆ 🚩

Edit ↻ ↺ | 🌐 ⌵ 🗑️ ⌵ 🔄 ⌵

^ Variable

Name HOOPS\_ServerUrl

Value http://localhost

Default Value http://localhost

Change HOOPS\_ServerUrl to point to Hoops Server through configured proxy.

18. Click Edit.

19. In the **Value** field, add `http://localhost/InnovatorServer/hoops_server/11182`

where,

- Localhost – IP Address of the machine
- InnovatorServer- Innovator instance of the user
- 11182- Hoops Server port number





20. Click Done to save the changes.
21. Go to "HOOPS Server\server\node\Config.js".
22. For publicHostname parameter, make the following change:

publicHostname: "localhost/Your innovator instance/hoops\_server",

publicHostname: "localhost/InnovatorServer/hoops\_server",

23. Go to "HOOPS Server\server\node\lib\SpawnerExtension.js".
24. For getEndpoint( req, res )method,

Change:

"endpoint": `\${Utils\_1.getWsProtocol(this.extConfig.sslEnableScServer)}://\${hostname}:\${this.extConfig.spawnServerPort}`,

To:

"endpoint": `\${Utils\_1.getWsProtocol(this.extConfig.sslEnableScServer)}://\${hostname}/\${this.extConfig.spawnServerPort}`,

```

getEndpoint(req, res) {
  const hostname = Utils_1.getPublicHostname(this.extConfig.publicHostname, this.extConfig.ipVersion);
  const json = {
    "endpoint": `${Utils_1.getWsProtocol(this.extConfig.sslEnableScServer)}://${hostname}/${this.extConfig.spawnServerPort}`,
  };
  Utils_1.sendJsonResponse(req, res, 200, json);
}
exports.SpawnerExtension = SpawnerExtension;
//# sourceMappingURL=SpawnerExtension.js.map

```



# Installation Guide



## Introduction

### Purpose

This Installation Guide describes installing the Aras 3D Visualization (3DV) 35 platform component.



## Scope

The installation guide provides requirements, Aras 3DV license information, detailed installation instructions, and troubleshooting information.



## Target Audience

This document is intended for administrators installing the Aras 3D Visualization platform component.



## Aras 3DV Architecture Overview

Aras 3D Visualization is modular and flexible.

The modularity is the Aras 3DV delivery as a set of the following components:

- **Aras CAD Converter:** This is a generator of viewable files that converts native 3D CAD files to formats such as PDF, XML, SCS, SCZ, PNG, and PRC by default.
- **Aras 3D Viewers:** This is a core 3D Visualization functionality that is the Monolithic 3D Viewer and a prerequisite for the Dynamic 3D Viewer.
- **Aras Dynamic Visualization:** The Dynamic 3D Viewer that includes the Dynamic Product Navigation (shattered viewing) and configuration capabilities for the 3D and Tree Grid Views based on selected Query and Dynamic View Definitions. It requires Aras 3D Viewers.
- **Aras Streaming Viewer (Hoops Server):** The Hoops Server provides the ability to use HOOPS Communicator server-side streaming for geometry data.

The flexibility is the possibility to install one, many, or all Aras 3D Visualization components on one or multiple machines per an organization's needs.

For more information on different viewers in Aras 3DV, please refer to *Aras 3D Visualization 35 - Administrator Guide*.



## Software and Hardware Requirements

### General Aras 3DV Requirements

Aras 3DV is an optional component for the Aras Innovator platform. It requires a certified or supported version of Aras Innovator with Conversion Server and Agent Service to be installed and running. For the Conversion Server requirements, see the Aras CAD Converter Requirements subsection.

At the time of its release, Aras 3D Visualization 35 was certified on Aras Innovator versions from 20 to 35. Given that newer Aras Innovator versions will be after this release, refer to the Support Matrix on the Subscriber Portal for the complete list of the certified or supported Aras Innovator versions.

Subscribers can obtain an Aras Innovator CD Image from the [Aras FTP](#) site. The installation and configuration documentation for a given Aras Innovator instance is in the Documentation folder of the Aras Innovator CD Image.

If using an earlier, unsupported Aras Innovator version, please upgrade it before installing Aras 3D Visualization. For upgrade assistance, consult Aras Support at [support@aras.com](mailto:support@aras.com) before installing Aras 3D Visualization.

For the complete list of required software, refer to the *Platform Support Matrix* section in the *Aras 3D Visualization 35 - Release Notes*.



## Aras CAD Converter Requirements

The Aras CAD Converter has software requirements in addition to the general Aras 3DV requirements. This required software should be installed and configured on the same machine where the CAD Converter will be installed.

These requirements can be excluded if the Aras CAD Converter will not be installed.

## Aras Innovator Conversion Server

The Aras CAD Converter uses the Aras Innovator Conversion Server that should be installed on one of the following 64-bit operating systems:

- Windows Server 2012
- Windows Server 2012 R2
- Windows Server 2016
- Windows Server 2019
- Windows Server 2022

The 32-bit operating systems are not supported.

## Microsoft Visual C++ Redistributable Package

The Aras CAD Converter and HOOPS Communicator may require the following redistributable packages to be installed on a machine with the Conversion Server, depending on the configuration:

- Microsoft Visual C++ 2010 64-bit
- Microsoft Visual C++ 2012 64-bit
- Microsoft Visual C++ 2013 64-bit
- Microsoft Visual C++ 2015 64-bit
- Microsoft Visual C++ 2017 64-bit

These packages can be downloaded from the Microsoft portal page with the latest supported Visual C++ downloads.

## Conversion Server Requirements and Recommendations

See the Aras Innovator 35 - Platform Specifications for the Conversion Server requirements and recommendations.



## Conversion Server and Agent Service Setup

The Conversion Server and Agent Service should be set up to work with Aras Innovator. For more information, refer to the Aras Innovator 35 – Conversion Server Setup Guide.



## Aras 3DV Licenses

The Aras 3D Visualization components require the following Feature Licenses:

- Aras CAD Converter: **Aras.CADConverter3**
- Aras 3D Viewers: **Aras.SecureSocial**

To request an activation key for a necessary Feature License, a customer should have an active Aras subscription and send an email to either:

- [licenses@aras.com](mailto:licenses@aras.com)
- Their account representative

The email should be in the following format:

- Subject: Feature License Name Activation Key Required
- Body: Aras Innovator Version

A reply should contain the Feature License activation key.

### Important

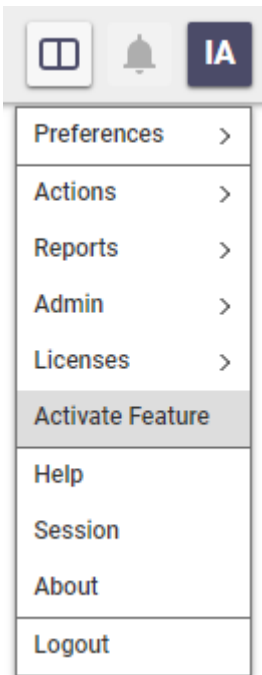
This distribution includes npm which is made available to you pursuant to the Artistic License 2.0. You may obtain the source for npm here: <https://github.com/npm/cli>.



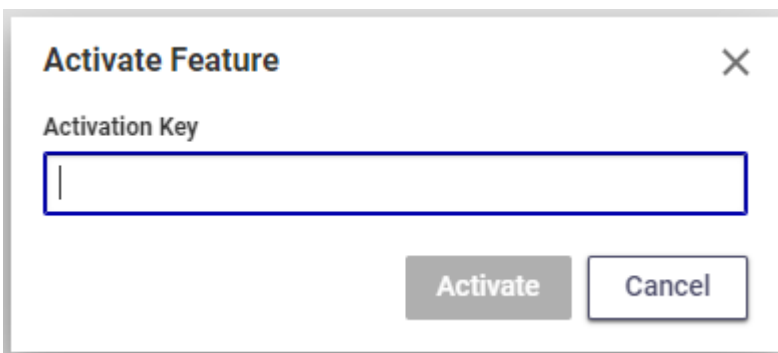
## CAD Converter License Activation

The following steps outline the process of activating the CAD Converter 3 Feature License:

1. Log into Aras Innovator as an administrator.
2. From the **User** menu near the upper-right corner, select **Activate Feature**.



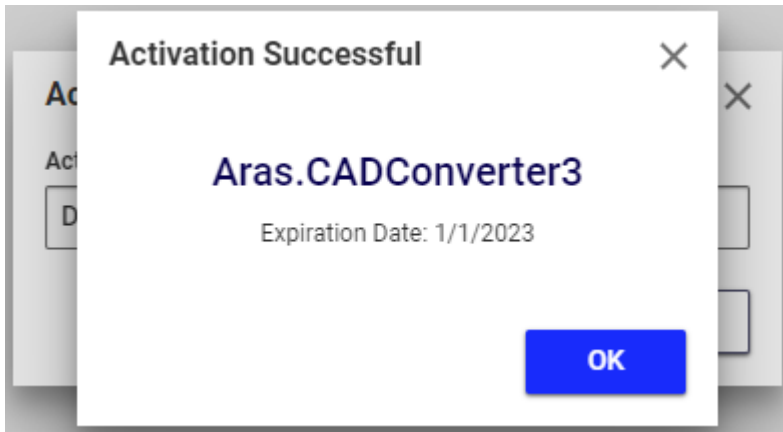
The **Activate Feature** dialog appears.



3. Enter the feature license activation key in the **Activation Key** field.
4. Click **Activate**.



The **Activation Successful** dialog appears.



Close the **Activate Feature** and **Activation Successful** dialogs.

## Download Aras 3DV

The Aras 3D Visualization 35 platform component is delivered as the Aras 3D Visualization 35 CD Image package, which subscribers can download from the [Aras File Sharing](#) site. The site content and folder structure are specific for a given logged-in user. The downloaded package may be a zip archive that should be extracted. The unzipped package should be copied to a machine where an Aras 3D Visualization component will be installed.



## Aras 3D Visualization Installation

The Aras 3D Visualization can be installed using the standard Aras tools.

Aras 3DV components should be installed in a specific order, except when all Aras 3DV components are simultaneously installed on the same machine.

All components can be installed on one machine if that machine has the Conversion and Aras Innovator Servers.

The Aras 3D Visualization installation modifies the Aras Innovator database and Conversion Server. Aras Innovator will be down during the installation.

Requirements:

- Aras Innovator Release 20 - 35
  - Aras Update Tool 1.23+
- or
- Aras Innovator Package Import Export Utilities.

### **Important**

Administrators who install 3D Visualization should ensure that all folders containing executable files (e.g, DLLs, EXE, etc.) are protected from unauthorized access

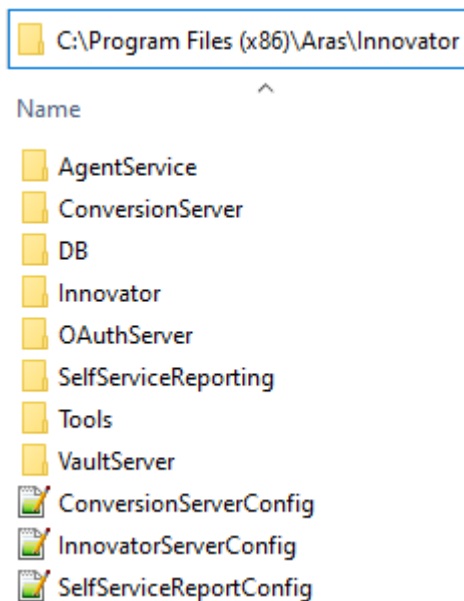


## Aras Innovator Code Tree

The Aras Innovator code tree includes files and folders installed on a disk with the first Aras Innovator installation.

The default code tree installation path is: `C:\Program Files(x86)\Aras\Innovator`

The following figure shows the out-of-the-box code tree content.



The following steps outline the process of locating the installation path if the Aras Innovator was installed in a different location:

1. Click the Windows **Start** button and go to **Control Panel**.
2. Select **Programs and Features**.
3. Click **Aras Innovator** from the installed applications list. The complete installation path appears in the Comments field.

Programs and Features

Control Panel > Programs > Programs and Features

Control Panel Home

Uninstall or change a program

To uninstall a program, select it from the list and then click Uninstall, Change, or Repair.

- View installed updates
- Turn Windows features on or off
- Install a program from the network

Name	Publisher	Installed On	Size	Version
Aras Innovator	Aras Corporation	6/15/2022	943 MB	14.0.7
Aras Office Connector	Aras Corporation	2/9/2022	11.8 MB	12.0.5.1957
Aras Update	Aras Corporation	5/6/2022	7.14 MB	1.17.1264

Aras Corporation Product version: 14.0.7 Support link: <http://www.ArasCorp...> Help link: <http://www.ArasCorp...> Size: 943 MB Comments: Installed to C:\Program Files (x86)\Aras\Innovator\



## Notification and Backup

1. Notify users that the system will be down at a scheduled time, and they should log out of the system before the start of the process.

### Important

It is best to give at least 24-hour notice, as well as a reminder 15 minutes before the upgrade.

2. Back up the code tree.

The **Code Tree** refers to files and folders installed on the disk when Aras Innovator was first installed.

The default path for the Code Tree installation would be something like:

`C:\Program Files (x86)\Aras\Innovator`

Name	Type	Size
AgentService	File folder	
ConversionServer	File folder	
DB	File folder	
Innovator	File folder	
OAuthServer	File folder	
SelfServiceReporting	File folder	
Tools	File folder	
VaultServer	File folder	
ConversionServerConfig	XML File	1 KB
InnovatorServerConfig	XML File	2 KB
SelfServiceReportConfig	XML File	1 KB
VaultServerConfig	XML File	1 KB

The default Conversion Server installation path is the following:

`C:\Program Files(x86)\Aras\ConversionServer`

3. Disconnect all users from the database.

The easiest way to prevent client sessions from committing any further changes to the database is to change the database connection string in the `InnovatorServerConfig.xml` from `<DB-Connection...` to `<xDB-Connection...`



and restart the **w3svc service (IIS)**. This expires all sessions and prevents all new connections to the Aras Innovator database through the existing instance.

4. Backup the database.

Store these files in a safe location, as they will need to be restored if the process fails.

5. Enable database connections.

After the database backup has been completed, enable the database connection string in the

`InnovatorServerConfig.xml` by changing it from `<xDB-Connection...` to `<DB-Connection...` and restarting the **w3svc service (IIS)**.



## Component Installation Order

Using Aras Update, all three Aras 3DV components (CAD Converter, 3D Viewer, and Dynamic Visualization) can be installed simultaneously on the same machine with the Conversion Server and the Aras Innovator Server. If otherwise, the components should be installed strictly in the following order:

1. Aras CAD Converter
2. Aras 3D Viewers
3. Aras Dynamic Visualization: Aras Dynamic Visualization requires Aras 3D Viewers.
4. Aras Streaming Viewer (Hoops Server)

Dynamic Visualization and Streaming Visualization use different view files (SCS and SCZ, respectively) and are incompatible with the different Viewers. Therefore, to install the Streaming Viewer, the user must select the **Use streaming SCZ by default** checkbox. To see the Streaming Viewer installation, please refer to the *Automated Aras 3DV Installation for Streaming Viewer* section.



## Installation of Dynamic 3D Viewer After Using Monolithic 3D Viewer

If installing the Dynamic 3D Viewer (Aras Dynamic Visualization) in an environment where CAD assemblies were viewed with the Monolithic 3D Viewer, the Dynamic Enable process should be activated for these CAD assemblies because Aras CAD Converter has not generated data for viewing them in the Dynamic 3D Viewer.

For process details and the activation procedure, see the *Dynamic Enabling* section in the *Aras 3D Visualization 35– Administrator Guide*.



## Installation of Streaming 3D Viewer After Using Monolithic 3D Viewer or Dynamic Viewer

### Important

Please note that the output files of the Dynamic/Monolithic Viewer and Streaming Viewer are incompatible. Suppose the Streaming 3D Viewer is installed in an environment where the Monolithic or Dynamic 3D Viewers have been previously installed and used. In that case, all existing native files of existing CAD Documents need to be re-converted. There is no automated means to perform this reconversion.

### Important

Only one Streaming Viewer can be installed on one machine at a time.

### Important

The Streaming Viewer currently cannot be deployed in a cloud environment. The HOOPS Server must be deployed with networked file access to a single vault containing view files for rendering.



## Automated Aras 3DV Installation

Aras Update provides an automated application installation process.

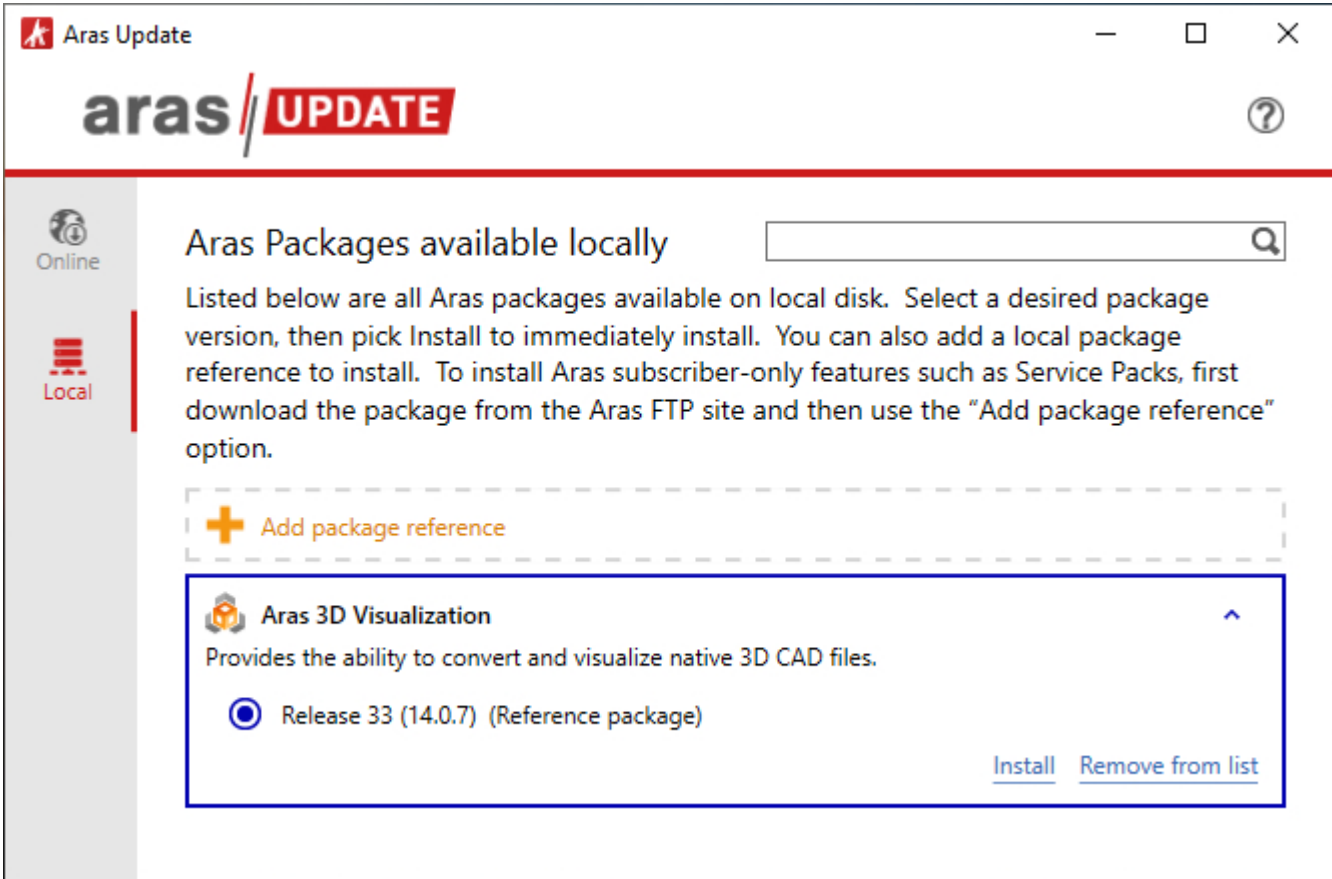
The installation procedure is the same when installing one, multiple, or all Aras 3D Visualization components on one machine. If multiple or all components are installed on multiple machines, the procedure should be performed on each machine in the order discussed in the *Component Installation Order* section.

Only a machine with the Conversion Server and Aras Innovator Server can install all components at once.

The following steps outline the process to install one, multiple, or all Aras 3D Visualization components on one machine using Aras Update:

1. Download the **Aras 3D Visualization 35 CD Image** from the Aras File sharing site and unzip the file on the local computer.
2. Enable the **Super User (root)** login.
3. Launch the **ArasUpdate.exe** file as an administrator.  
If default options were selected during installation, the **ArasUpdate.exe** file is found in the directory `C:\Program Files (x86)\Aras\Aras Update\`.
4. On the **Local** tab, click **Add package reference** and navigate to the folder where the **Aras 3D Visualization** package was extracted in the first step.
5. Expand the **Aras 3D Visualization** package, select **Release 35 (14.0.8)** and click **Install**.





6. Select the installation modules of the Aras 3D Visualization components that will be installed and click **Next**. Deselect the **Hoops Server** component. This component is used to install Streaming Viewer. To install Streaming Viewer, see the *Automated Aras 3DV Installation for Streaming Viewer* section.

### Important

The Aras CAD Converter, Aras 3D Viewer, and Aras Dynamic Visualization components consist of two installation modules: code tree and database updates. Both modules should be installed to install a component; otherwise, the component will not be completely installed. Updating the code tree and database in separate installation sessions is possible.





Aras 3D Visualization Release 33 (14.0.7) requires the following components for proper installation. Select the components you intend to install.

Computer name: ARAS-CORP/AC5CG3221VKB

- Aras CAD Converter**  
Provides the ability to convert native 3D CAD files for use with 3D Viewers.
- Aras CAD Converter Database Updates**  
Import to Innovator Database.
- Aras 3D Viewers**  
Provides core 3D Visualization functionality for Monolithic and Dynamic 3D Viewers.
- Aras 3D Viewers Database Updates**  
Import to Innovator Database.
- Aras Dynamic Visualization**  
Provides the ability to configure 3D and Tree Grid Views for the Dynamic 3D Viewer based on selected Query and Dynamic View Definitions.
- Aras Dynamic Visualization Database Updates**  
Import to Innovator Database.
- Hoops Server**  
Provides the ability to use HOOPS Communicator server-side streaming for geometry data.

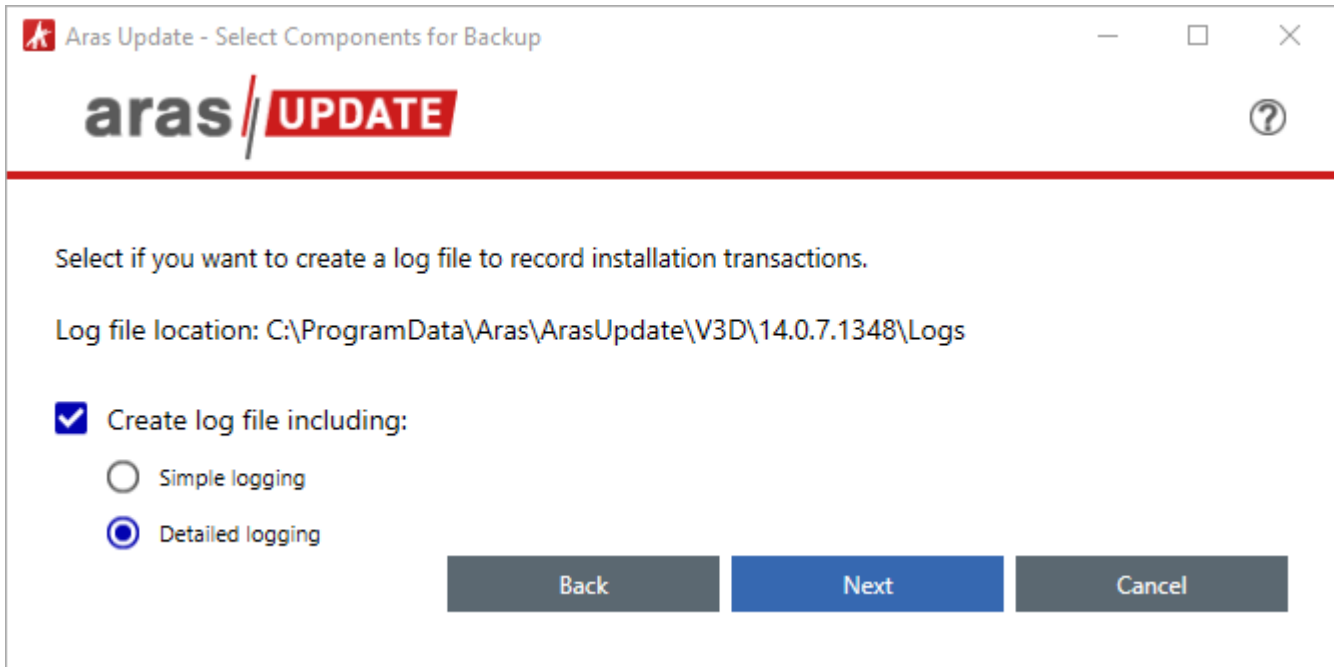
Back

Next

Cancel

7. Select the **Detailed logging** option and click **Next**.





The logging option records the installation attempt and can be used to troubleshoot issues.

8. Input the connection information and click **Next**.



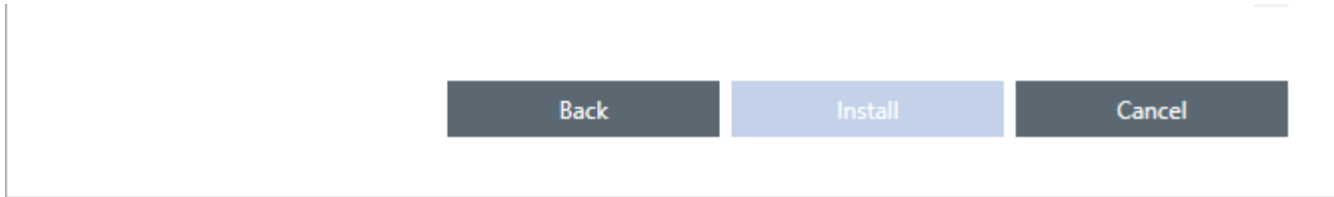
Aras Update

# aras // UPDATE

Enter the following parameters for each component you are going to install and then click Install button to continue.

- Aras CAD Converter**
  - Path to root folder where Innovator Server is installed
  - Specify local directory where HOOPS Converter should be installed (example: 'C:\HOOPS Converter')
  - Use streaming SCZ by default
- Aras CAD Converter Database Updates**
  - Server URL (example - 'http://localhost/InnovatorServer')
  - Database Name (example - 'InnovatorSolutions')
  - Username (should use 'root' User)
  - Password for 'root' User
- Aras 3D Viewers**
- Aras 3D Viewers Database Updates**
- Aras Dynamic Visualization**
- Aras Dynamic Visualization Database Updates**





- **Path to root folder where Aras Innovator Server is installed:** The path to the Aras Innovator code tree. See the *Aras Innovator Code Tree* section.
- **Specify local directory where HOOPS Converter should be installed:** The physical path to a local folder where HOOPS Converter files will be installed; for example, `C:\HOOPS Converter`.

### Important

When installing the Streaming Viewer, the Use streaming SCZ by default field should be checked; otherwise, leave it unchecked.

### Important

If the **Use streaming SCZ by default** checkbox is checked, and the Streaming Service (HOOPS Server) is not installed, none of the other Viewers will work.

- **Server URI:** The URI for connecting to a given Aras Innovator instance. By default, it is `http://localhost/InnovatorServer/`.
- **Database Name:** The name of a target Aras Innovator database. By default, it is `InnovatorSolutions`.
- **Username:** `root`
- **Password for 'root' User:** The password for the root User. By default, it is `innovator`.

If installing multiple Aras 3D Visualization components, connection settings for the following components are automatically inherited from the preceding ones. For example, if Aras 3D Viewers is installed with Aras CAD Converter, the Aras 3D Viewers modules inherit connection settings from the Aras CAD Converter modules.

9. Once the connection information is provided, click Install. The installation process begins.



Installation is in progress. Please, wait...



Installing...

[Show log file](#)

Cancel

10. Once the installation is successfully completed, close Aras Update.





Installation completed successfully!

[Show log file](#)

Home

Exit

11. Disable the **Super User (root)** login.
12. This step is optional. Confirm the successful Aras 3D Visualization installation. See the *Confirming Aras 3DV Installation* section.

If the installation fails, restore the Aras Innovator code tree, Conversion Server, and database with the backups done before the installation and contact Aras Support at [support@aras.com](mailto:support@aras.com) .



## Automated Aras 3DV Installation for Streaming Viewer

### Important

Please note that the output files of Dynamic/Monolithic Viewer and Streaming Viewer are incompatible with one another. If the Streaming 3D Viewer is installed in an environment where the Monolithic or Dynamic 3D Viewers have been previously installed and used, all existing native files of existing CAD Documents need to be re-converted. There is no automated means to perform this reconversion.

### Important

Only one Streaming Viewer can be installed on one machine at a time.

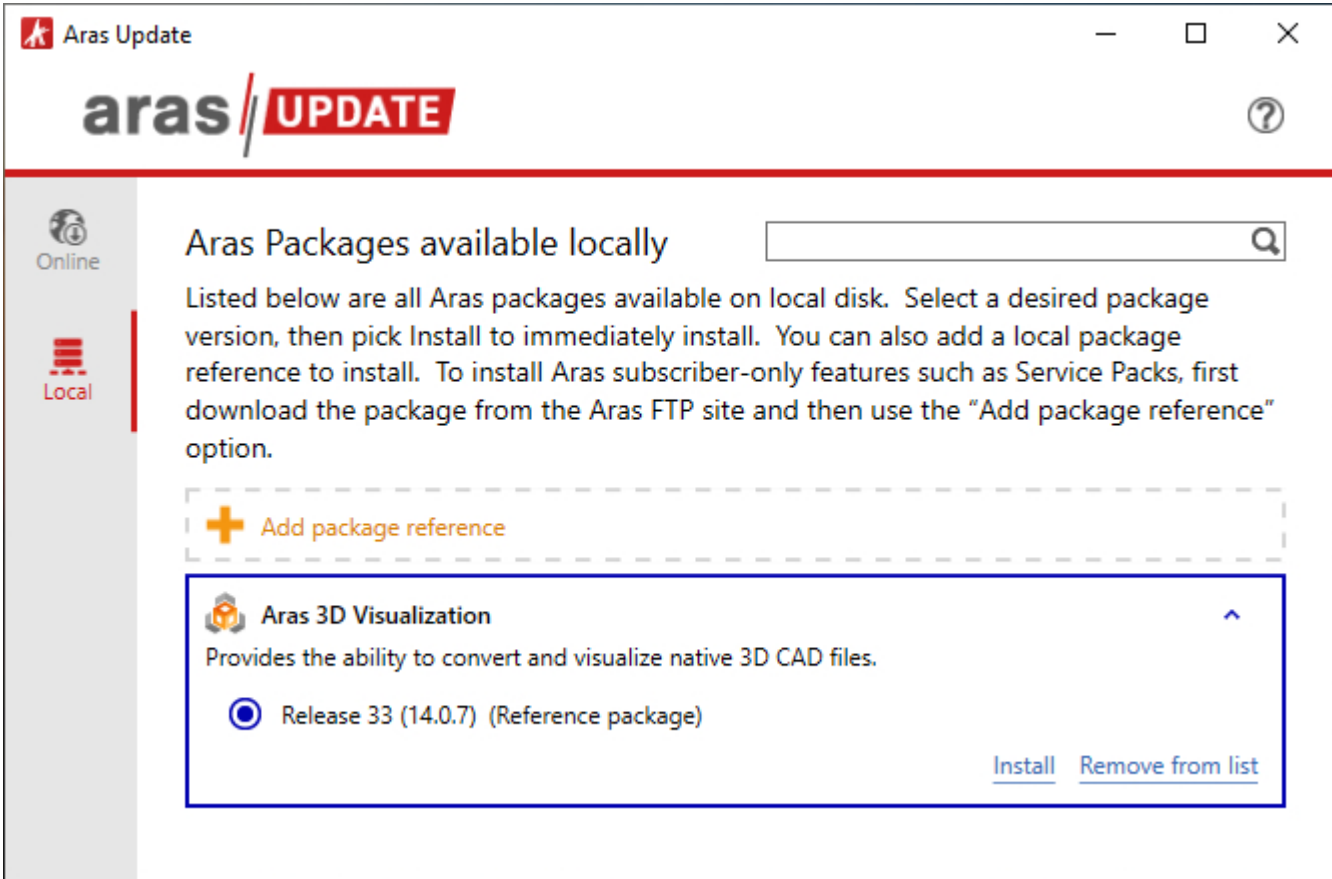
### Important

The Streaming Viewer currently cannot be deployed in a cloud environment. The HOOPS Server must be deployed with networked file access to a single vault containing view files for rendering.

The following steps outline the process of Aras 3DV installation for Streaming Viewer:

1. Download the **Aras 3D Visualization 35 CD Image** from the Aras File sharing site, and unzip the file on the local computer.
2. Enable the **Super User (root)** login.
3. Please confirm that the **Aras.CADConverter3** license is activated.
4. Launch the **ArasUpdate.exe** file as an administrator.  
If default options were selected during installation, the **ArasUpdate.exe** file is found in the directory `C:\Program Files (x86)\Aras\Aras Update\`.
5. On the **Local** tab, click **Add package reference** and navigate to the folder where the **Aras 3D Visualization** package was extracted in the first step.
6. Expand the **Aras 3D Visualization** package and click **Install**.





7. Select the installation modules of the Aras 3D Visualization components that will be installed on the given machine and click **Next**.



Aras 3D Visualization Release 33 (14.0.7) requires the following components for proper installation. Select the components you intend to install.

Computer name: ARAS-CORP/AC5CG3221VKB

- Aras CAD Converter**  
Provides the ability to convert native 3D CAD files for use with 3D Viewers.
- Aras CAD Converter Database Updates**  
Import to Innovator Database.
- Aras 3D Viewers**  
Provides core 3D Visualization functionality for Monolithic and Dynamic 3D Viewers.
- Aras 3D Viewers Database Updates**  
Import to Innovator Database.
- Aras Dynamic Visualization**  
Provides the ability to configure 3D and Tree Grid Views for the Dynamic 3D Viewer based on selected Query and Dynamic View Definitions.
- Aras Dynamic Visualization Database Updates**  
Import to Innovator Database.
- Hoops Server**  
Provides the ability to use HOOPS Communicator server-side streaming for geometry data.

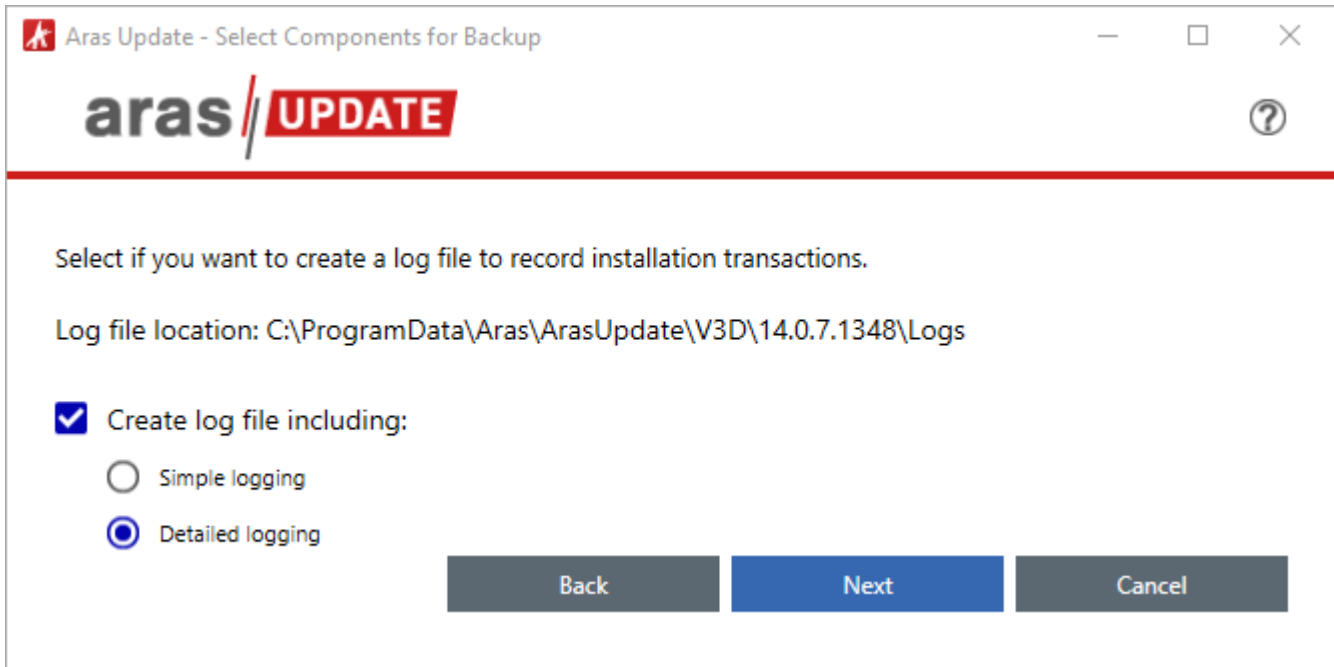
Back

Next

Cancel

8. Select the **Detailed logging** option and click **Next**.





The logging option records the installation attempt and can be used to troubleshoot issues.

9. Specify the connection information for the installation modules of the components to be installed:



Aras Update

# aras UPDATE

Enter the following parameters for each component you are going to install and then click Install button to continue.

- Aras CAD Converter**

Path to root folder where Innovator Server is installed

Specify local directory where HOOPS Converter should be installed (example: 'C:\HOOPS Converter')

Use streaming SCZ by default

- Aras CAD Converter Database Updates**

Server URL (example - 'http://localhost/InnovatorServer')

Database Name (example - 'InnovatorSolutions')

Username (should use 'root' User)

Password for 'root' User

- Aras 3D Viewers
- Aras 3D Viewers Database Updates
- Aras Dynamic Visualization
- Aras Dynamic Visualization Database Updates
- Hoops Server



Path to the folder where HOOPS Server should be installed

Hostname of the machine where HOOPS Server should be installed

Path to the vault folder for corresponding Innovator instance

### Component CAD Converter Updates:

- **Path to root folder where Aras Innovator Server is installed:** The physical path to the Aras Innovator code tree. See the *Aras Innovator Code Tree* section.
- **Specify local directory where HOOPS Converter should be installed:** The physical path to a local folder where HOOPS Converter files will be installed; for example, `C:\HOOPS Converter`.

#### Important

When installing the Streaming Viewer, the **Use streaming SCZ by default** field should be checked otherwise leave it unchecked.

#### Important

If the **Use streaming SCZ by default** checkbox is checked, and the Streaming Service is not installed, none of the other viewers will work.

### Component Database Updates:

- **Server URI:** The URI for connecting to a given Aras Innovator instance. By default, it is `http://localhost/InnovatorServer/`.
- **Database Name:** The name of a target Aras Innovator database. By default, it is `InnovatorSolutions`.
- **Username:** `root`
- **Password for 'root' User:** The password for the root User. By default, it is `innovator`.

### Hoops Server:



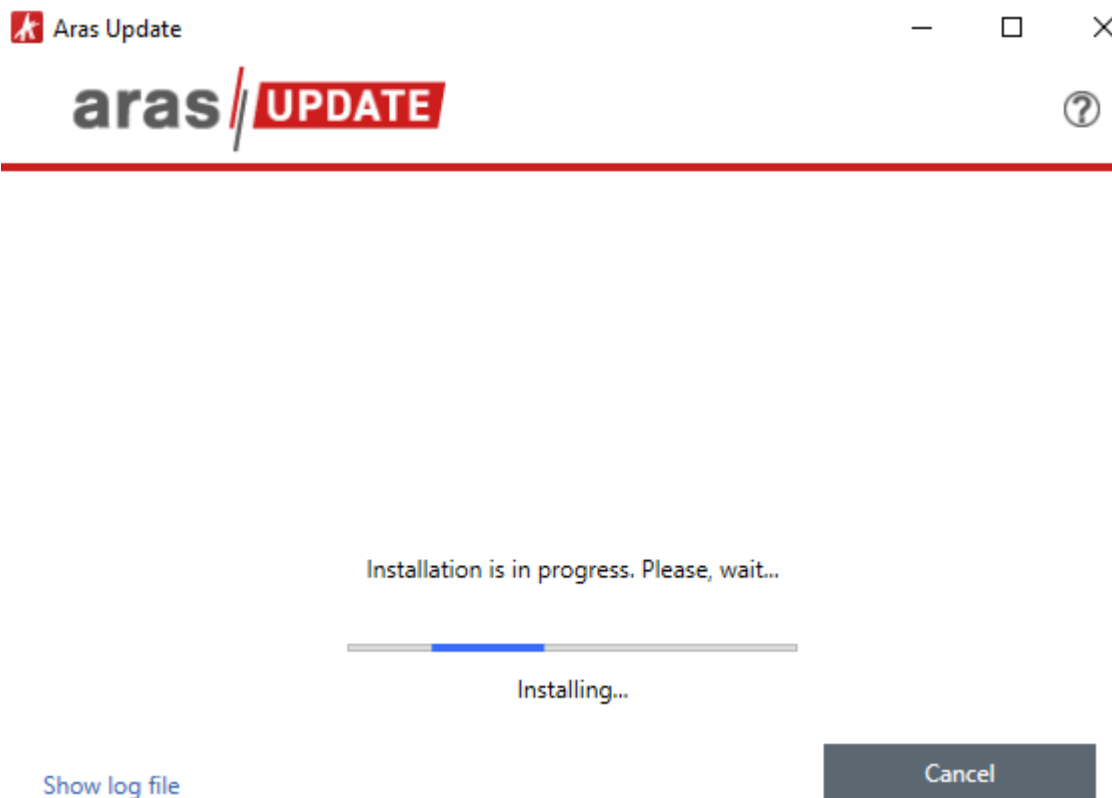
- **Path to the folder where HOOPS Server should be installed:** The physical path to a local folder where HOOPS Server files will be installed; for example, `C:\HOOPS_Server`.
- **Hostname of the machine where HOOPS Server should be installed:** The hostname of a local machine where HOOPS Server will be installed.
- **Path to the vault folder corresponding Aras Innovator instance:** The physical path to the vault folder of the Aras Innovator instance.

### Important

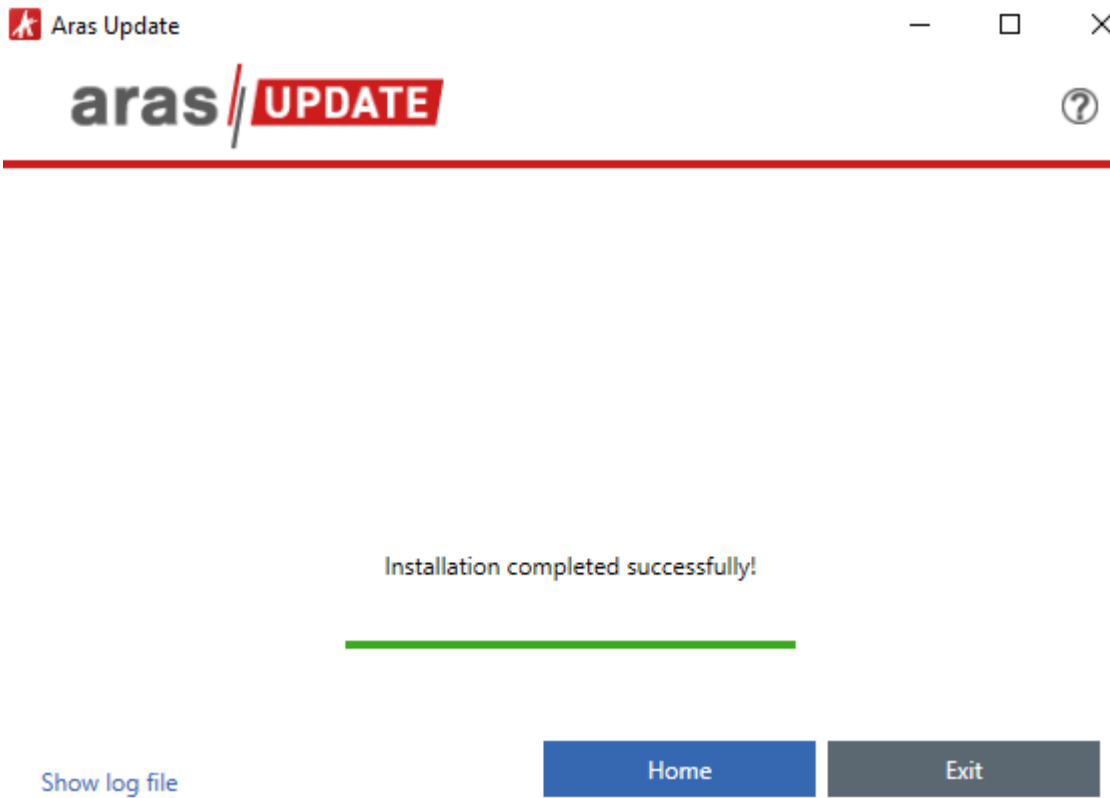
The Streaming Viewer is supported only with a single vault. The Streaming Viewer service must be installed on a machine with a direct network access to the vault.

If multiple Aras 3D Visualization components are installed, connection settings for the following components are automatically inherited from the preceding ones. For example, if Aras 3D Viewers is installed with Aras CAD Converter, the Aras 3D Viewers modules inherit connection settings from the Aras CAD Converter modules.

10. Once the connection information is provided, click **Install**. The installation process begins.

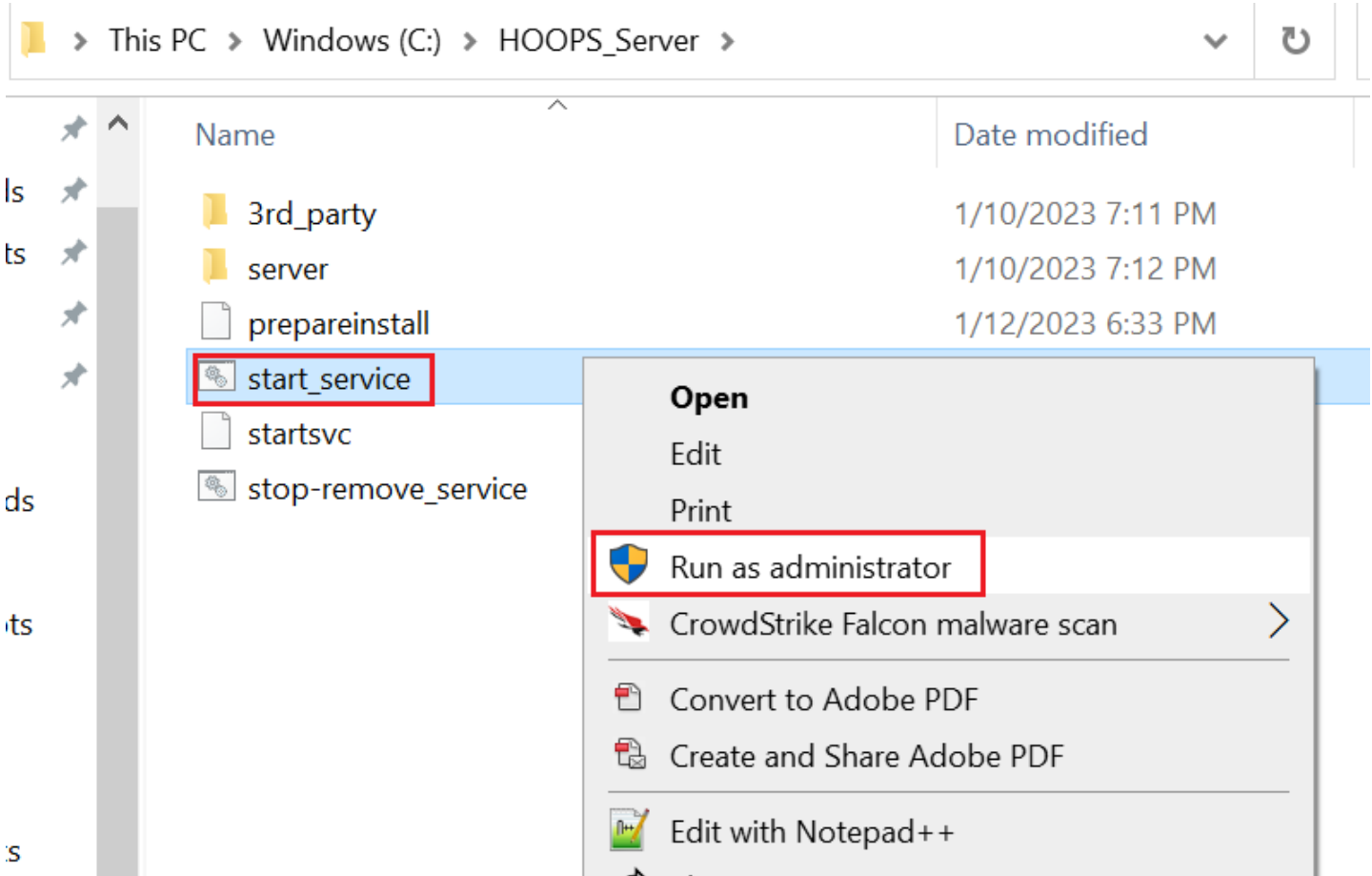


11. Once the installation is successfully completed, close Aras Update.



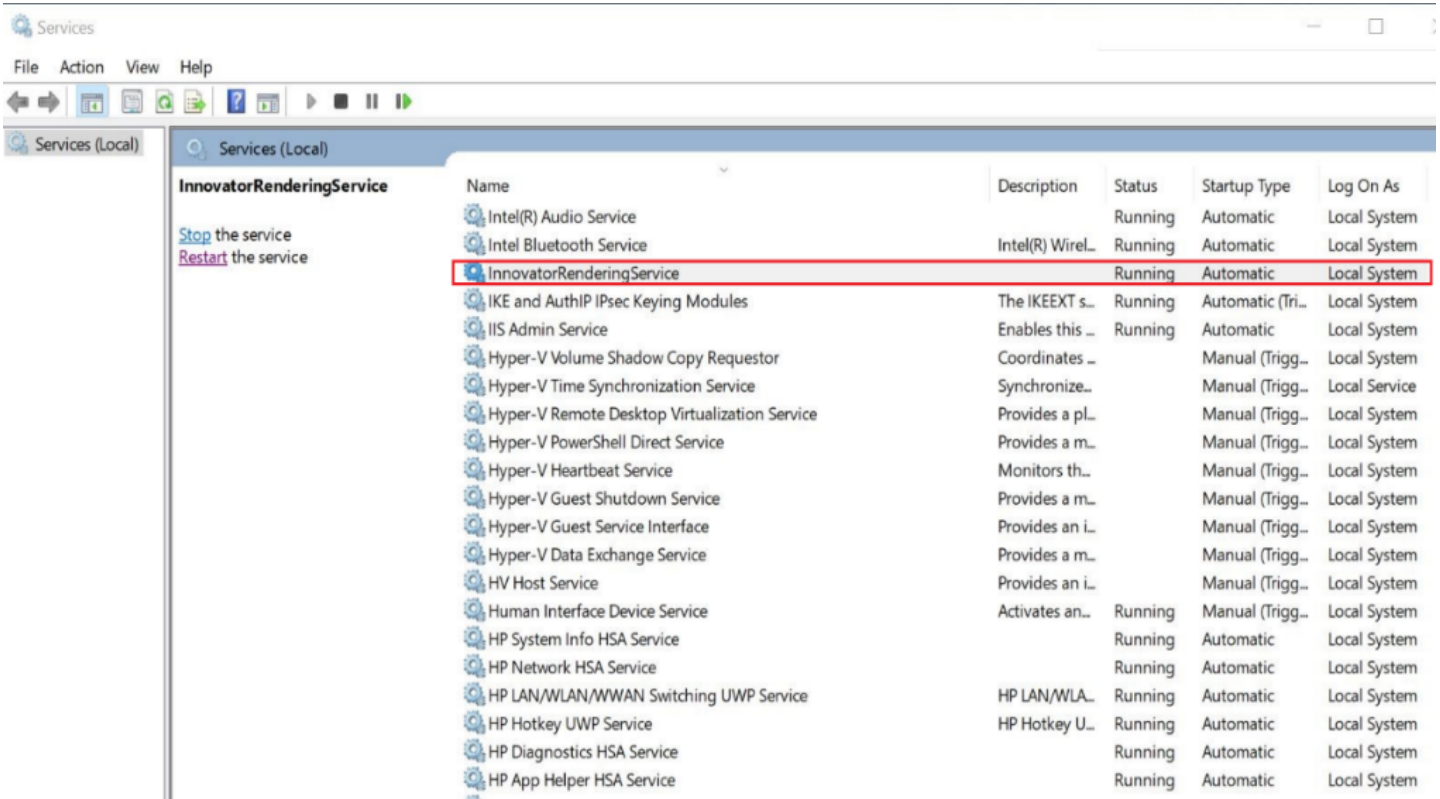
12. On the C drive in the **Hoops Server** file, right-click **start\_service** and select **Run as Administrator**.





13. Open Windows **Services** and ensure that **InnovatorRenderingService** is added and running.





14. Disable the **Super User (root)** login.
15. For the required setup to get the streaming viewer running, refer to the *Administrative Configuration for Streaming Viewer* section.
16. This step is optional. Confirm the successful Aras 3D Visualization installation. See the *Confirming Aras 3DV Installation* section.

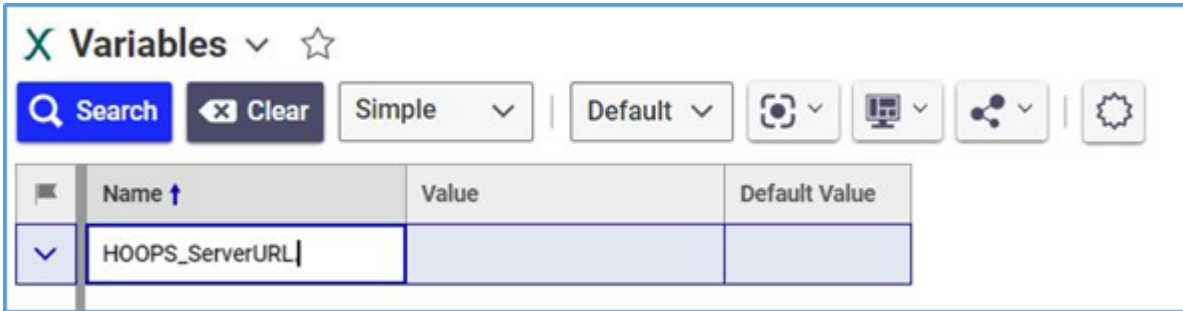
If the installation fails, restore the Aras Innovator code tree, Conversion Server, and database with the backups done before the installation and contact Aras Support at [support@aras.com](mailto:support@aras.com) .

### Administrative Configurations for Streaming Viewer

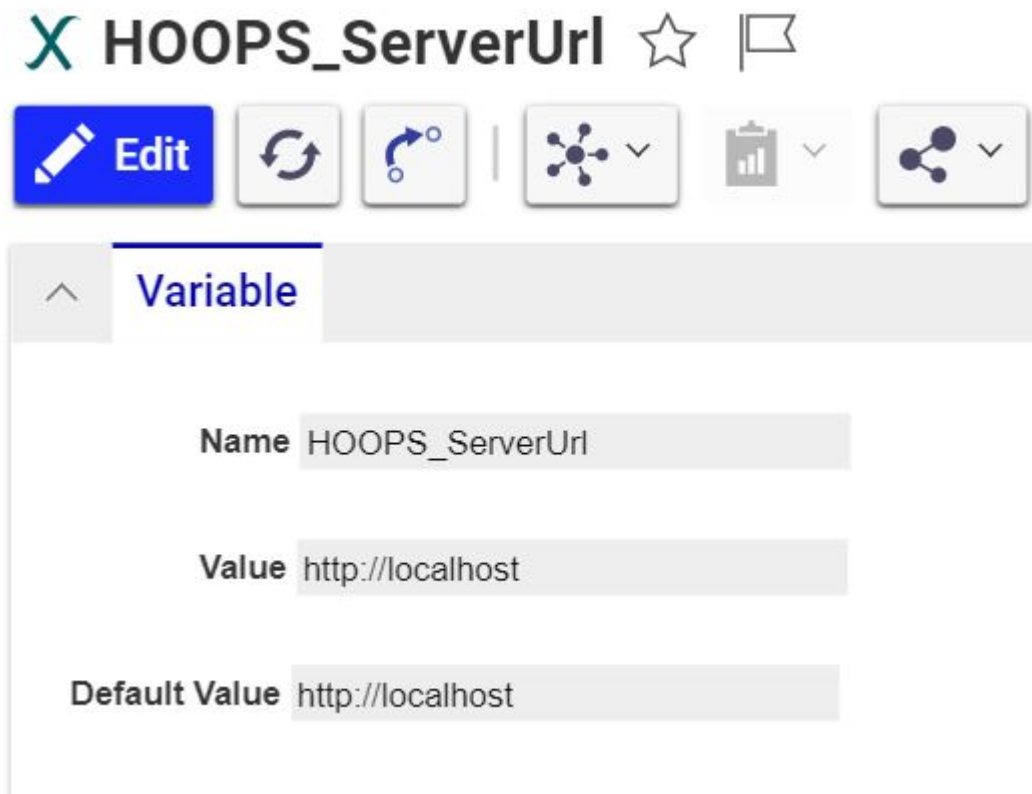
The following steps outline the administrative setup to get the Streaming Viewer running:

1. Login into the Aras Innovator as **Administrator**.
2. From **Table of Contents**, expand **Administration** and select **Variables**.
3. Click **Search Variables** and search for **HOOPS\_ServerURL**.





The `HOOPS_ServerURL` form appears.



4. In **Value** field, replace `http://localhost` with [http://IP\\_Address:11182](http://IP_Address:11182)

where,

- **IP\_Address**: IP Address of the machine where the Streaming Viewer is installed.
- **11182**: Default Hoops Server Port Number.



X HOOPS\_ServerUrl ☆ 🚩

Save Done Discard ↻

Variable

Name HOOPS\_ServerUrl

Value http://10.6.75.157:11182

Default Value http://localhost

**Important**

The port number 11182 in HOOPS\_ServerURL should be the same as the spawnServerPort parameter value in the HOOPS Server\server\node\Config.js.

There is no change to the Default Value field.

5. Click **Done** to save the changes.

**Important**

If the server where Streaming Viewer is installed has Graphic Processing Unit (GPU), then the **windowsServiceRespawnEnabled** parameter in the Config.js file should be set to **False**.



## Manual Aras 3DV Installation

Aras Package Import provides a manual application installation process.

The installation procedure differs across Aras 3D Visualization components. The subsections of this section discuss the procedure for each component. The installation procedure should be performed for every component to be installed in the order discussed in the *Component Installation Order* section.

## Manual Aras CAD Converter Installation

The following steps outline the process of manual Aras CAD Converter Installation:

1. In the unzipped Aras 3D Visualization 35 CD Image package, navigate to the following folder: Aras 3D Visualization 35 CD Image\Packages\CADConverter.
2. Copy the HOOPS Converter folder from the CADConverter package folder into a permanent location on a machine with the Conversion Server, for example, C:\HOOPS Converter.
3. Copy the ConversionServer folder from the CADConverter package folder to a folder that includes the ConversionServer folder with the installed Conversion Server.
4. Open the ConversionServerConfig.xml file in a text editor with an admin's privileges. This file is the root Aras Innovator code tree folder for the default installation. Do not confuse this file with the ConversionServer.config XML file in the ConversionServer folder with the Conversion Server.
5. In the child `sectionGroup name="ConverterSettings"` tag of the `configSections` tag, add the following section tags with attributes:

```
<configSections>
<sectionGroup name="ConverterSettings">
<section name="ArasCadConverter"
type="Aras.ConversionFramework.Converter.Hoops.Configuration.HoopsConverterConf
ArasCadConverter"></section>
<section name="ArasCadConverterPrc"
type="Aras.ConversionFramework.Converter.Hoops.Configuration.HoopsConverterConf
ArasCadConverter"></section> <section name="DpnCadConverterStepJt"
type="Aras.ConversionFramework.Converter.Hoops.Configuration.HoopsConverterConf
ArasCadConverter" />
```



```
</sectionGroup>
```

```
</configSections>
```

These new tags define the configurations of the new `ArasCadConverter` and `ArasCadConverterPrc` converters.

- In the child `Converters` tag of the `ConversionServer` tag, add the following `Converter` tags with attributes:

```
<ConversionServer>
```

```
<Converters>
```

```
<Converter name="Aras CAD to PDF Converter"
type="Aras.ConversionFramework.Converter.Hoops.HoopsConverter,
ArasCadConverter" />
```

```
<Converter name="Aras PRC to SCS Converter"
type="Aras.ConversionFramework.Converter.Hoops.HoopsConverterPrc,
ArasCadConverter" />
```

```
<Converter name="JT Step CAD Converter"
type="Aras.CadConverter.StepJtCadConverter, ArasCadConverter"
xdt:Transform="Insert"/>
```

```
</Converters>
```

```
</ConversionServer>
```

These new tags define the new converters. Such a tag specifies a DLL for a given converter and a Class within this DLL that performs the conversion.

- In the `ConverterSettings` tag, add the following `ArasCadConverter` and `ArasCadConverterPrc` tags with child tags and attributes:

```
<ConverterSettings>
```

```
<ArasCadConverter>
```

```
<Application converterPath="C:\HOOPS Converter\bin\converter.exe"/>  
  
<Command arguments="--sc compute_bounding_boxes 'All' --  
input pdf template file 'C:\HOOPS  
Converter\templates\Blank Template L.pdf' --output pdf  
'%filepath%\%filename%.pdf' --output png '%filepath%\%filename%.png' --  
output png resolution '150x150' --output scs '%filepath%\%filename%.scs' -  
-output xml assemblytree '%filepath%\%filename%.xml' --output prc  
'%filepath%\%filename%.prc' --background color '1.0, 1.0, 1.0' --  
output_logfile '%filepath%\%filename%.log'" />  
  
<Output>  
  
<UploadToVault>  
  
<File extension="prc" argsMarkers="--output_prc" />  
<File extension="scs" argsMarkers="--output_scs" />  
<File extension="pdf" argsMarkers="--output_pdf" />  
<File extension="png" argsMarkers="--output_png" />  
<File extension="stl" argsMarkers="--output_stl" />  
<File extension="xml" argsMarkers="--output_xml_assemblytree" />  
  
</UploadToVault>  
  
</Output>  
  
<AssemblyCommand dynamicEnabled="True" arguments="--  
sc compute bounding boxes 'All' --input pdf template file 'C:\HOOPS  
Converter\templates\Blank Template L.pdf' --output pdf  
'%filepath%\%filename%.pdf' --output png '%filepath%\%filename%.png' --  
output png resolution '150x150' --output scs '%filepath%\%filename%.scs' -  
-output_xml_assemblytree '%filepath%\%filename%.xml' --output prc
```



```
'%filepath%\%filename%.prc' --background color '1.0, 1.0, 1.0' --  
output_logfile '%filepath%\%filename%.log' />  
  
</ArasCadConverter>  
  
<ArasCadConverterPrc>  
  
<Application converterPath="C:\H00PS Converter\bin\converter.exe"/>  
  
<Command arguments="--output scs '%filepath%\%filename%.scs' --  
output xml assemblytree '%filepath%\%filename%.xml' --output_logfile  
'%filepath%\%filename%.log' " />  
  
<Output>  
  
<UploadToVault>  
  
<File extension="prc" argsMarkers="--output_prc" />  
<File extension="scs" argsMarkers="--output_scs" />  
<File extension="pdf" argsMarkers="--output_pdf" />  
<File extension="png" argsMarkers="--output_png" />  
<File extension="stl" argsMarkers="--output_stl" />  
<File extension="xml" argsMarkers="--output_xml_assemblytree" />  
  
</UploadToVault>  
  
</Output>  
  
</ArasCadConverterPrc>  
<DpnCadConverterStepJt>  
  
<Application converterPath="/opt/exchange/bin/linux64"/>  
  
<Parameters>
```



```

<Parameter key="JTVersion" value="8.1" /> <!-- Optional. Possible values:
8.1|9.5|10.0 -->
</Parameters>
</DpnCadConverterStepJt>
</ConverterSettings>

```

The `ArasCadConverter` and `ArasCadConverterPrc` tags set up the new converters.

The `converterPath` attributes should include the path to the `converter.exe` application in the HOOPS Converter folder copied in the third step.

The `--input_pdf_template_file` argument in the `arguments` attributes should include the path to the `Blank_Template_L.pdf` template file in the HOOPS Converter folder copied in the third step.

An entire `<... arguments="..." />` attribute must be only one single line without carriage returns or newlines.

The child `File` tags of the `UploadToVault` tags enable the files of the given file types to be uploaded to the Vault and rendered correctly by the Viewer.

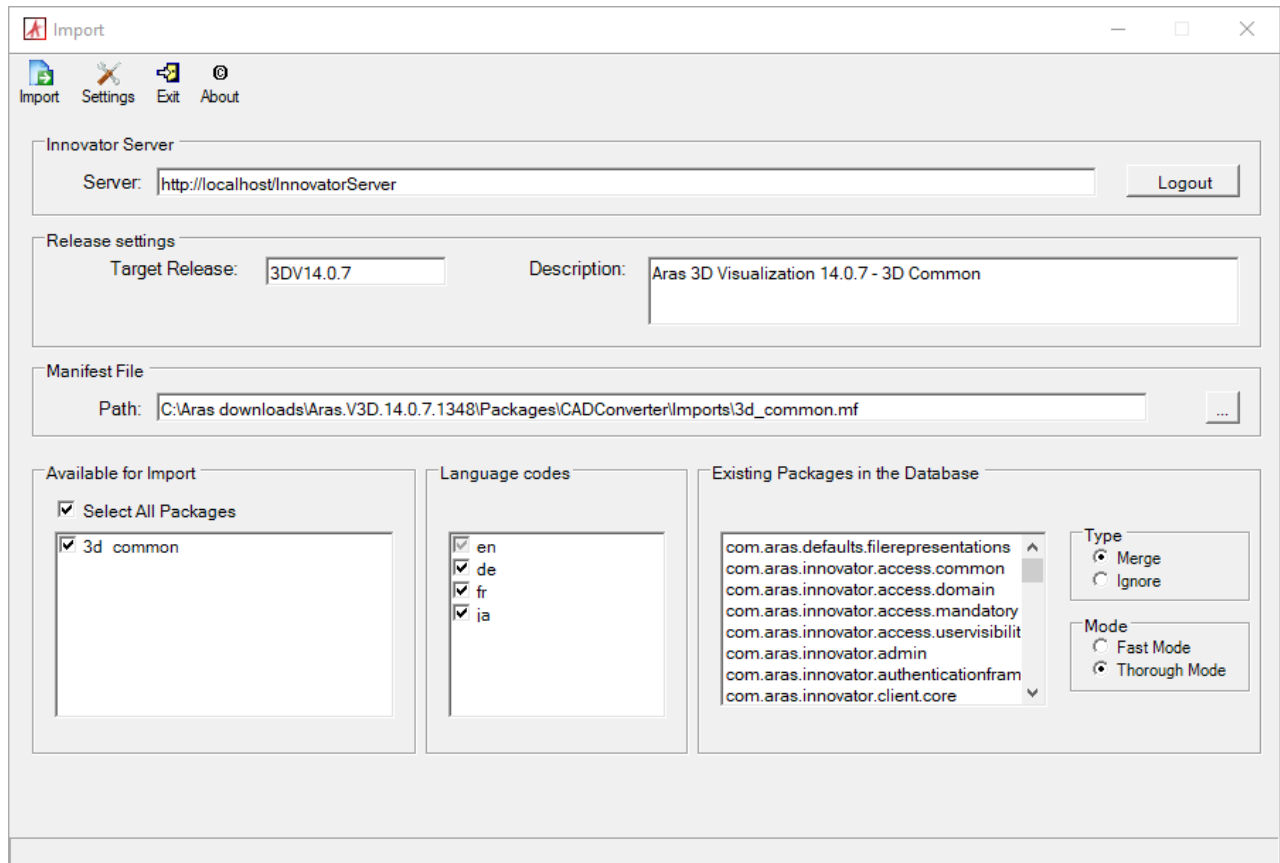
Capitalization differences, typos, extra newlines, and carriage returns in the `ConversionServerConfig.xml` file raise an error discussed in the 500.19 Internal Server Error section.

8. Save and close the `ConversionServerConfig.xml` file.
9. Restart the **Internet Information Services (IIS)**.
10. Import the Aras 3D Visualization 3D Common database package with the **Package Import Export Utilities**. For more information using this tool, refer to the *Aras Innovator 35 – Package Import Export Utilities* documentation.

1. Enable the **Super User (root)** login.



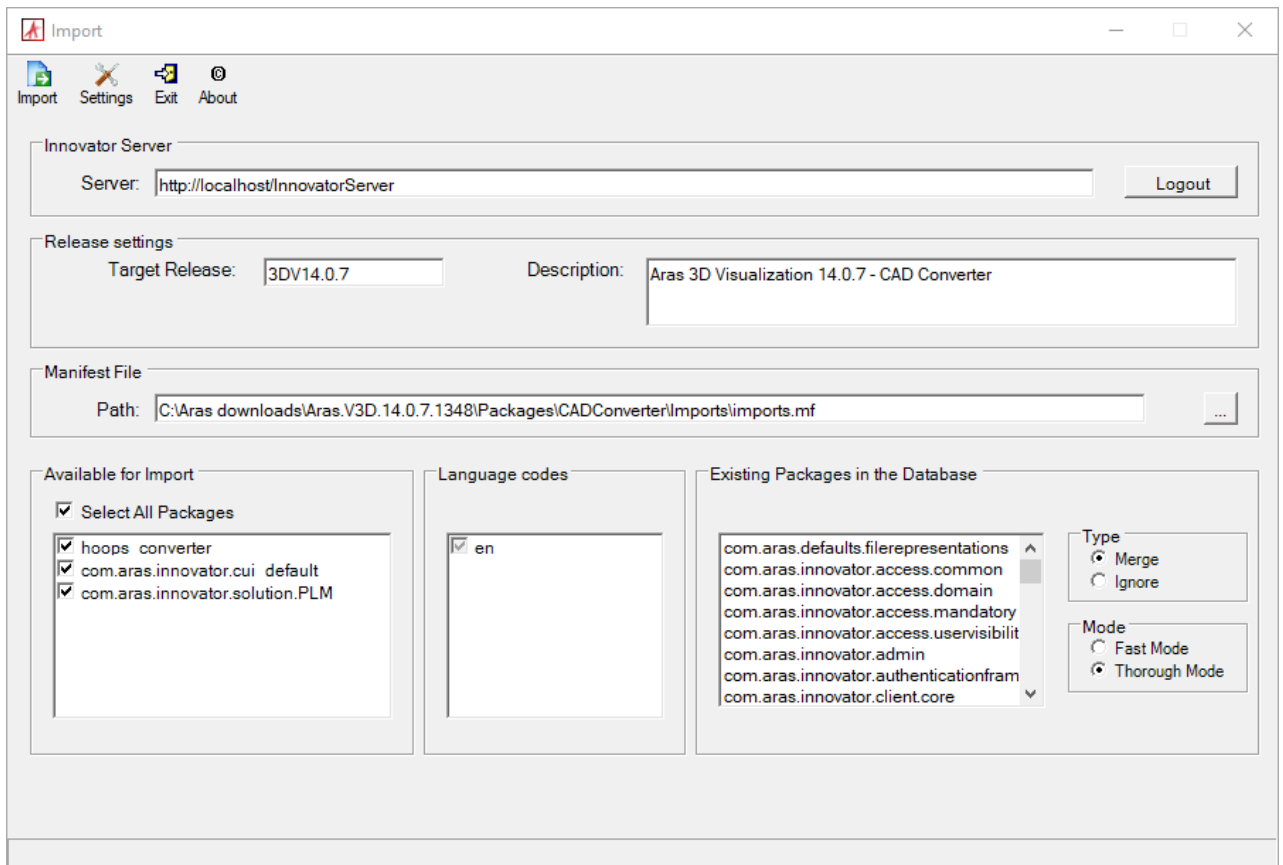
- Browse to the `\PackageImportExportUtilities\Import\` folder and run the **Import.exe** file.



- Input the connection information.
  - Server:** The connection URL for Aras Innovator. By default, it is `http://localhost/InnovatorServer/`.
  - Click the **Login** button and enter:
    - Database:** The target Aras Innovator database. By default, it is `InnovatorSolutions`.
    - Username:** `root`
    - Password:** Password for “root” login (Default is “innovator”)
  - Target Release:** `3DV14.0.8`
  - Description:** `Aras 3D Visualization 14.0.8 – 3D Common`
  - Manifest File Path:** The manifest file `\Packages\CADConverter\Imports\3d_common.mf`
  - Available for Import:** Select All Packages
  - Type:** Merge
  - Mode:** Thorough Mode
- Click the **Import** button.

11. Import the Aras 3D Visualization CAD Converter database package with the **Package Import Export Utilities**. For more information using this tool, refer to the *Aras Innovator 35 – Package Import Export Utilities* documentation.

1. Enable the **Super User (root)** login.
2. Browse to the `\PackageImportExportUtilities\Import\` folder and run the **Import.exe** file.



3. Input the connection information.
  - **Server:** The connection URL for Aras Innovator. By default, it is `http://localhost/InnovatorServer/`. Click the **Login** button and enter:
    - **Database:** The target Aras Innovator database. By default, it is `InnovatorSolutions`.
    - **Username:** `root`
    - **Password:** Password for “root” login (Default is “innovator”)
  - **Target Release:** `3DV14.0.8`
  - **Description:** `Aras 3D Visualization 14.0.8 – CAD Converter`
  - **Manifest File Path:** The manifest file `\Packages\CADConverter\Imports\imports.mf`



- **Available for Import:** Select All Packages
  - **Type:** Merge
  - **Mode:** Thorough Mode
4. Click the **Import** button.

12. Disable the **Super User (root)** login.

### Important

The **Super User (root)** login should not be enabled in production.

13. This step is optional. Confirm the successful installation. See the *Confirming Aras 3DV Installation* section.
14. If the installation fails, restore the Aras Innovator code tree, Conversion Server, and database with the backups done before the installation and contact Aras Support at [support@aras.com](mailto:support@aras.com).

## Manual Aras 3D Viewers Installation

The following steps outline the process of manual Aras 3D Viewer:

1. In the unzipped Aras 3D Visualization 35 CD Image package, navigate to the following folder:  
`Aras 3D Visualization 35 CD Image\packages\3DViewers`
2. Copy the Innovator folder from the `3DViewers` package folder to the root Aras Innovator code tree folder where the `Innovator` folder exists.
3. Replace the files in the destination if it is prompted.

### Important

It is recommended that a server administrator performs this step.

4. In the Aras Innovator code tree, navigate to `\Innovator\Client` and open the `InnovatorClient.config` file in a text editor run as an administrator.
5. In the `cachingModule` tag, change the value of the `filesRevision` attribute from `std` to `2`. If it is already an integer, change it to a next higher value; for example, from `2` to `3`:

```
<configuration>
```

```
...
```

```
<cachingModule moduleEnabled="true" filesRevision="2" />
```

```
...
```



</configuration>

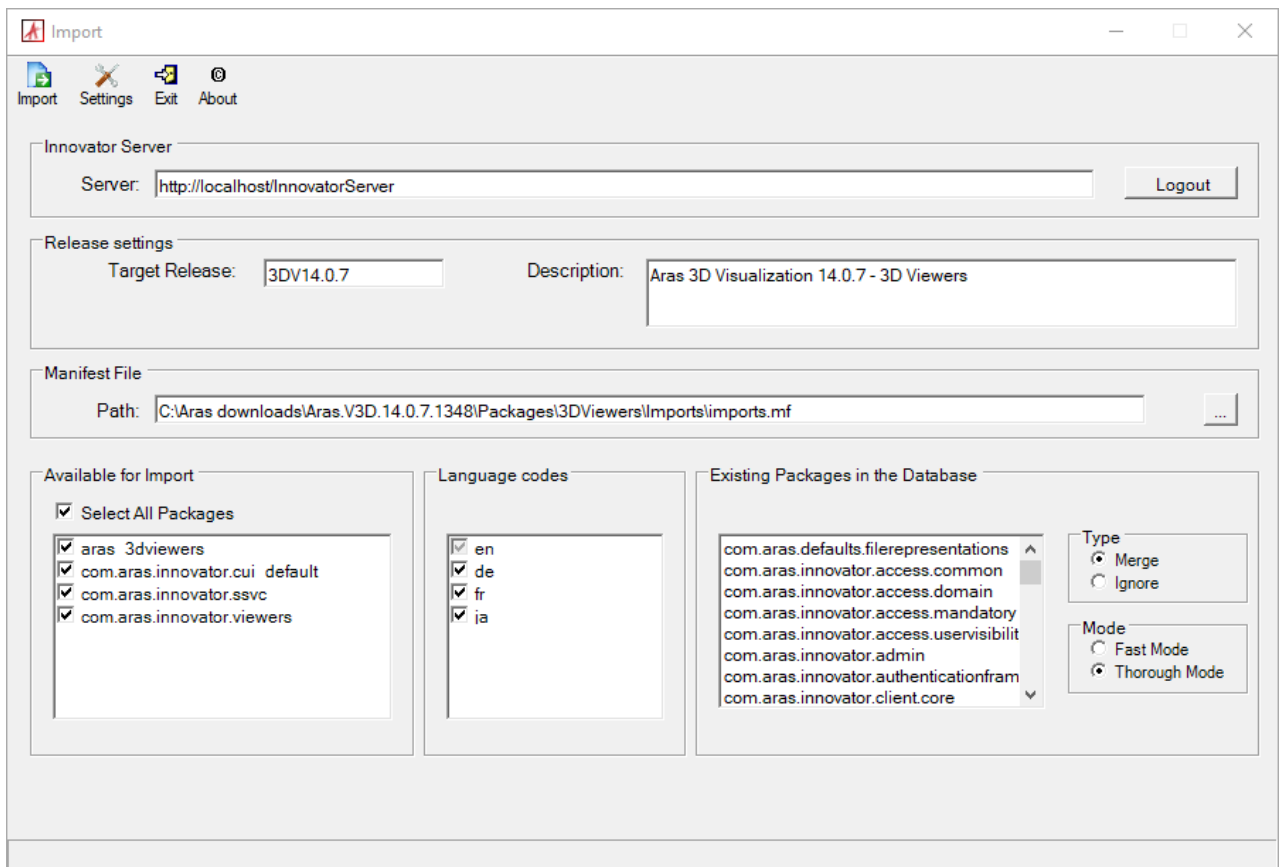
Such value modification reloads the cache. If the cache is not reloaded after the Aras 3DV installation, the Aras 3DV UI may not be displayed correctly; for example, icons can be absent.

6. Save and close the `InnovatorClient.config` file.

7. Import the Aras 3D Visualization 3D Viewers database package with the **Package Import Export Utilities**. For more information using this tool, refer to the Aras Innovator 35 – Package Import Export Utilities documentation.

1. Enable the **Super User (root)** login.

2. Browse to the `\PackageImportExportUtilities\Import\` folder and run the **Import.exe** file.



3. Input the connection information.



- **Server:** The connection URL for Aras Innovator. By default, it is `http://localhost/InnovatorServer/`.  
Click the **Login** button and enter:
  - **Database:** The target Aras Innovator database. By default, it is `InnovatorSolutions`.
  - **Username:** `root`
  - **Password:** Password for “root” login (Default is “innovator”)
- **Target Release:** `3DV14.0.8`
- **Description:** `Aras 3D Visualization 14.0.8 – 3D Viewers`
- **Manifest File Path:** The manifest file `\Packages\3DViewers\Imports\imports.mf`
- **Available for Import:** `Select All Packages`
- **Type:** `Merge`
- **Mode:** `Thorough Mode`

4. Click the **Import** button.

8. Disable the **Super User (root)** login.

### Important

The **Super User (root)** login should not be enabled in production.

9. This step is optional. Confirm the successful Aras 3D Viewers installation. See the [Confirming Aras 3DV Installation](#) section.
10. If the installation fails, restore the Aras Innovator code tree and database with the backups done before the installation and contact Aras Support at [support@aras.com](mailto:support@aras.com).

## Manual Aras Dynamic Visualization Installation

The following steps outline the process of Manual Aras Dynamic Visualization Installation:

1. In the unzipped Aras 3D Visualization 35 CD Image package, navigate to the following folder:  
`Aras 3D Visualization 35 CD Image\Packages\DPN`
2. Copy the Innovator folder from the DPN package folder to the root Aras Innovator code tree folder that includes the Innovator folder.
3. Replace the files in the destination if the system prompts.

### Important

It is recommended that a server administrator performs this step.

4. In the Aras Innovator code tree, navigate to `\Innovator\Server` and open the `method-config.xml` file in a text editor run as administrator
5. In the `ReferencedAssemblies` tag, add the following child name tags:



```
<ReferencedAssemblies>
```

```
...
```

```
<name>$(binpath)/Aras.DynamicModelViewer.Core.dll</name>
```

```
<name>$(binpath)/Aras.DynamicModelViewer.DataModel.dll</name>
```

```
<name>$(binpath)/Aras.DynamicModelViewer.QueryProcessor.dll</name>
```

```
<name>Microsoft.Extensions.Logging.dll</name>
```

```
</ReferencedAssemblies>
```

6. Save and close the `method-config.xml` file.

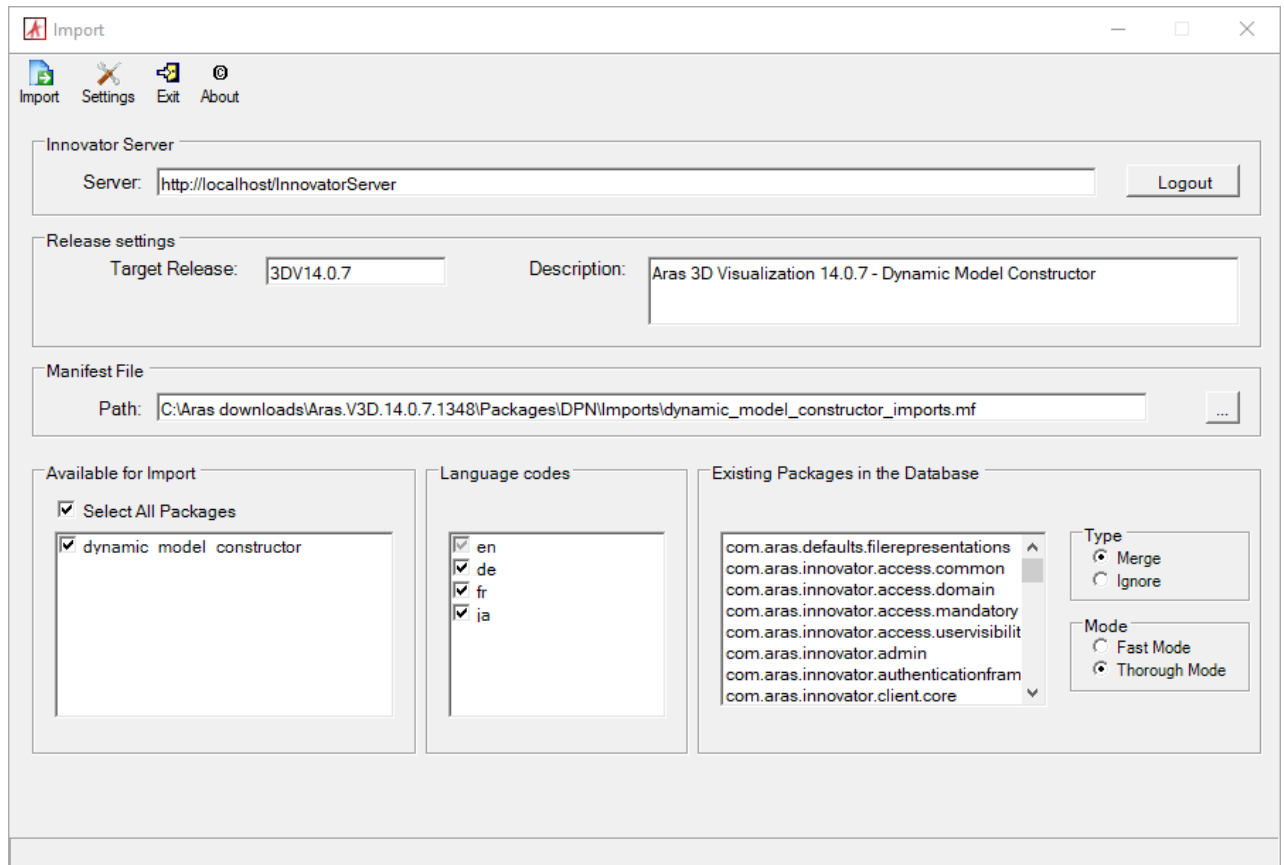
7. Restart the **Internet Information Services (IIS)**.

8. Import the Aras 3D Visualization Dynamic Model Constructor database package with the **Package Import Export Utilities**. For more information using this tool, refer to the *Aras Innovator 35 – Package Import Export Utilities* documentation.

1. Enable the **Super User (root)** login.

2. Browse to the `\PackageImportExportUtilities\Import\` folder and run the **Import.exe** file.





3. Input the connection information.

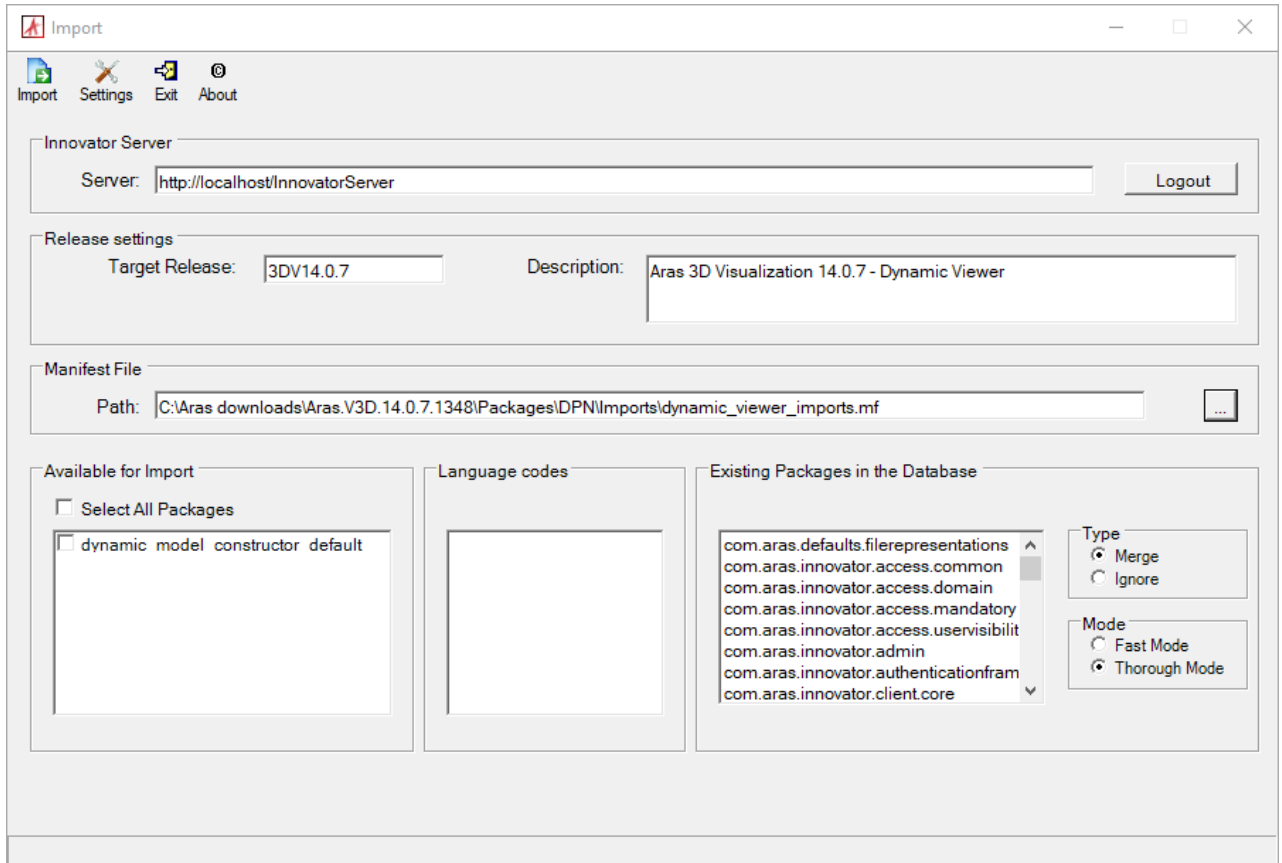
- **Server:** The connection URL for Aras Innovator. By default, it is `http://localhost/InnovatorServer/`. Click the **Login** button and enter:
  - **Database:** The target Aras Innovator database. By default, it is `InnovatorSolutions`.
  - **Username:** `root`
  - **Password:** Password for “root” login (Default is “innovator”)
- **Target Release:** `3DV14.0.8`
- **Description:** `Aras 3D Visualization 14.0.8 – Dynamic Model Constructor`
- **Manifest File Path:** The manifest file  
`\Packages\DPN\Imports\dynamic_model_constructor_imports.mf`
- **Available for Import:** Select All Packages
- **Type:** Merge
- **Mode:** Thorough Mode

4. Click the **Import** button.

9. Import the Aras 3D Visualization Dynamic Viewer database package with the **Package Import Export Utilities**. For more information using this tool, refer to the *Aras Innovator 35 – Package Import Export Utilities* documentation.



1. Enable the **Super User (root)** login.
2. Browse to the `\PackageImportExportUtilities\Import\ f` folder and run the **Import.exe** file.



3. Input the connection information.
  - **Server:** The connection URL for Aras Innovator. By default, it is `http://localhost/InnovatorServer/`. Click the **Login** button and enter:
    - **Database:** The target Aras Innovator database. By default, it is `InnovatorSolutions`.
    - **Username:** `root`
    - **Password:** Password for “root” login (Default is “innovator”)
  - **Target Release:** `3DV14.0.8`
  - **Description:** `Aras 3D Visualization 14.0.8 – Dynamic Viewer`
  - **Manifest File Path:** The manifest file `\Packages\DPN\Imports\dynamic_viewer_imports.mf`
  - **Available for Import:**  Select All Packages
  - **Type:** Merge
  - **Mode:** Thorough Mode
4. Click the **Import** button.



10. Disable the **Super User (root)** login.

#### Important

The **Super User (root)** login should not be enabled in production.

11. This step is optional. Confirm the successful Aras Dynamic Visualization installation. See the *Confirming Aras 3DV Installation* section.
12. If the installation fails, restore the Aras Innovator code tree and database with the backups done before the installation and contact Aras Support at [support@aras.com](mailto:support@aras.com).

## Manual Aras Streaming Viewer Installation

#### Important

Please note that the output files of Dynamic/Monolithic Viewer and Streaming Viewer are incompatible with one another. If the Streaming 3D Viewer is installed in an environment where the Monolithic or Dynamic 3D Viewers have been previously installed and used, all existing native files of existing CAD Documents need to be re-converted. There is no automated means to perform this reversion.

The Streaming Viewer currently cannot be deployed in a cloud environment. The HOOPS Server must be deployed with networked file access to a single vault containing view files for rendering.

#### Important

Only one Streaming Viewer can be installed on one machine at a time.

The following steps outline the process of manual installation of Aras 3DV Streaming Viewer:

1. In the unzipped **Aras 3D Visualization 35 CD Image package**, navigate to the following folder:  
Aras 3D Visualization 35 CD Image\Packages\HOOPS Server
2. Copy the **HOOPS Server** folder from **HOOPS Server** package to a permanent location on a machine. For example, C:\HOOPS Server.
3. In the **HOOPS Server** folder, go to C:\HOOPS Server\server\node and open **Config.js** file in a text editor run as administrator.
4. For **publicHostname** parameter, replace "localhost" value with the IP Address of the local machine. For example, **publicHostname: "10.188.182.38"**



```

28 var config = {
29   // The port for the spawn-server REST and proxy calls. Note that setting this to 0 or null will disable the
30   // the spawn-server, which may help with troubleshooting.
31   spawnServerPort: 11182,
32
33   // The hostname to use for broker-connection stream-cache servers or when enabling SSL.
34   // This field can be used to generate endpoints containing a public DNS or IP address.
35   // If not specified, the system will attempt to determine an appropriate value.
36   // This value has no effect when using proxying unless SSL is enabled.
37   // When SSL is enabled, this hostname must be verifiable against the supplied
38   // certificate chain.
39   publicHostname: "10.188.182.38",
40

```

5. For `communicatorDir` parameter, enter the path to `Hoops Server` directory created in the third step.  
For example, `communicatorDir: "C:/Hoops Server"`

```

121   // Points to the root of the communicator package, can be relative or absolute. If it's
122   // relative, then it's relative to the root of the server/node directory.
123   communicatorDir: "C:/HOOPS_Server",

```

6. For `modelDirs` parameter, enter the path to Aras Innovator instance Vault.

For example, `modelDirs: ["C:/Aras/Vault/14SP10", ]`

```

129   // Array of directories that contain the models available to the stream-cache servers.
130   // Any relative directories are relative to 'communicatorDir'
131   // Note that SCS models are not delivered by the spawn server, thus the paths are not included
132   modelDirs: ["C:/Aras/Vault/14SP10",
133     "./quick_start/converted_models/user/sc_models",
134     "./quick_start/converted_models/authoring_samples_data",
135     "./quick_start/converted_models/standard/sc_models",
136   ],

```

7. Set the `windowsServiceRespawnEnabled` parameter to true.

```

267   // See https://docs.microsoft.com/en-us/windows/win32
268   windowsServiceRespawnEnabled: true,

```

### Important

If the server where Streaming Viewer is installed has Graphic Processing Unit (GPU), then the `windowsServiceRespawnEnabled` parameter in the `Config.js` file should be set to false.

8. Copy the **Aras.CADConverter3** license from the Feature Licenses. Only copy the content from the `<HoopsLicense>` tag for the latest available version for example:

```

<license id="hoops2024">
<HoopsLicense>4E...xeR</HoopsLicense>

```



F72D3A13DB604BA1BBDBAC0AA9CAAADC ☆ |

Edit ↻ ↶ ⚙️ 🗑️ 🔗 ⋮

### Feature License

User Count	Named User List	Login

Server Name	Server Locale	Server MAC Address	Password
		0F:8A:4F:8A:05	...

DB Server Name	DB Server Engine Edition
	4

License Data

```
</license>
<license id="hoops2024">
<HoopsLicense>
[Redacted License Data]
</HoopsLicense>
```

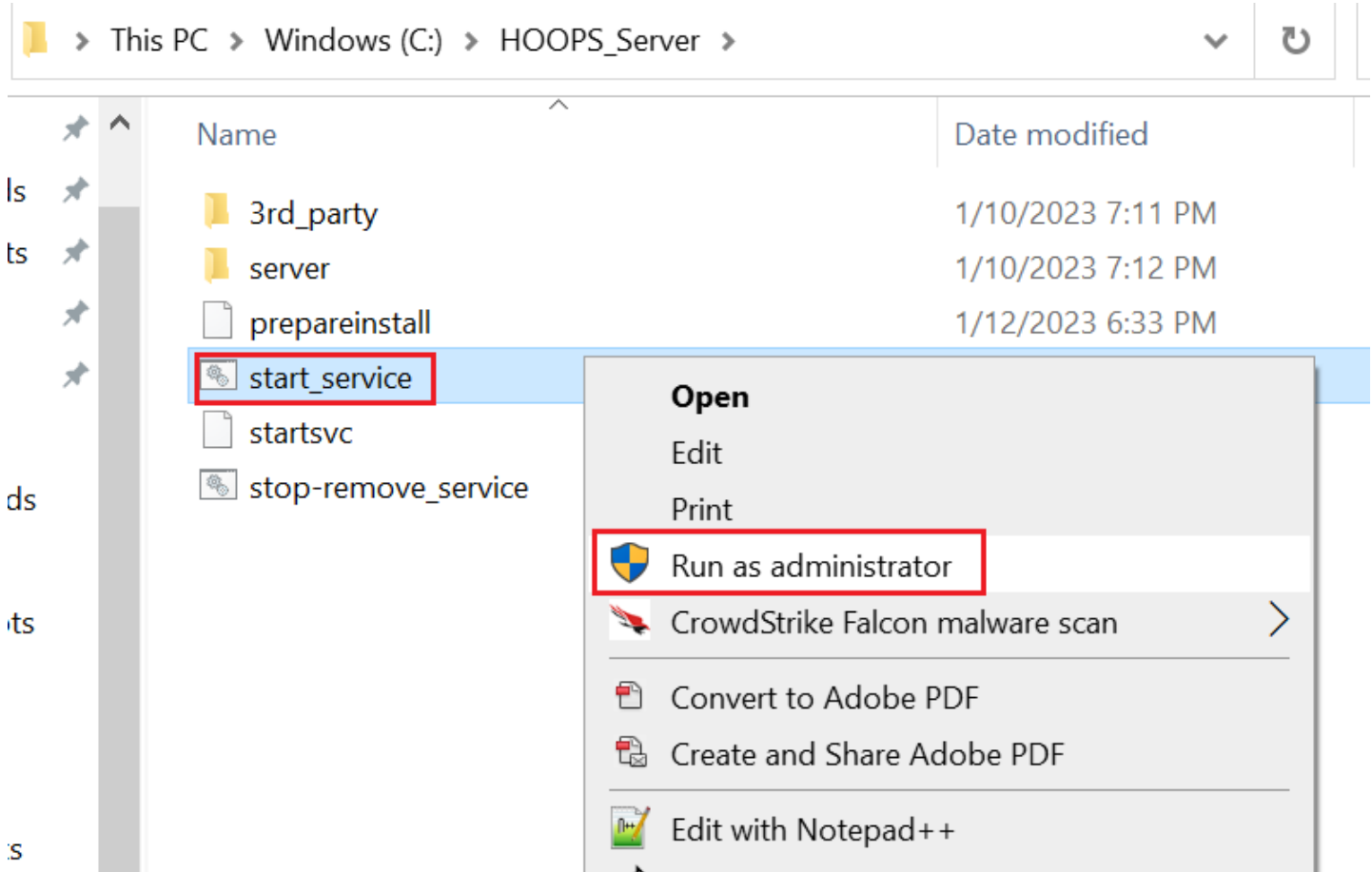
9. In the `Config.js` file, for the `license` parameter, replace it with the newest license copied from the previous step.

```
// A valid communicator license key, unless 'licenseFile' is defined, then null
license:
"
```

10. Save the `Config.js` file.

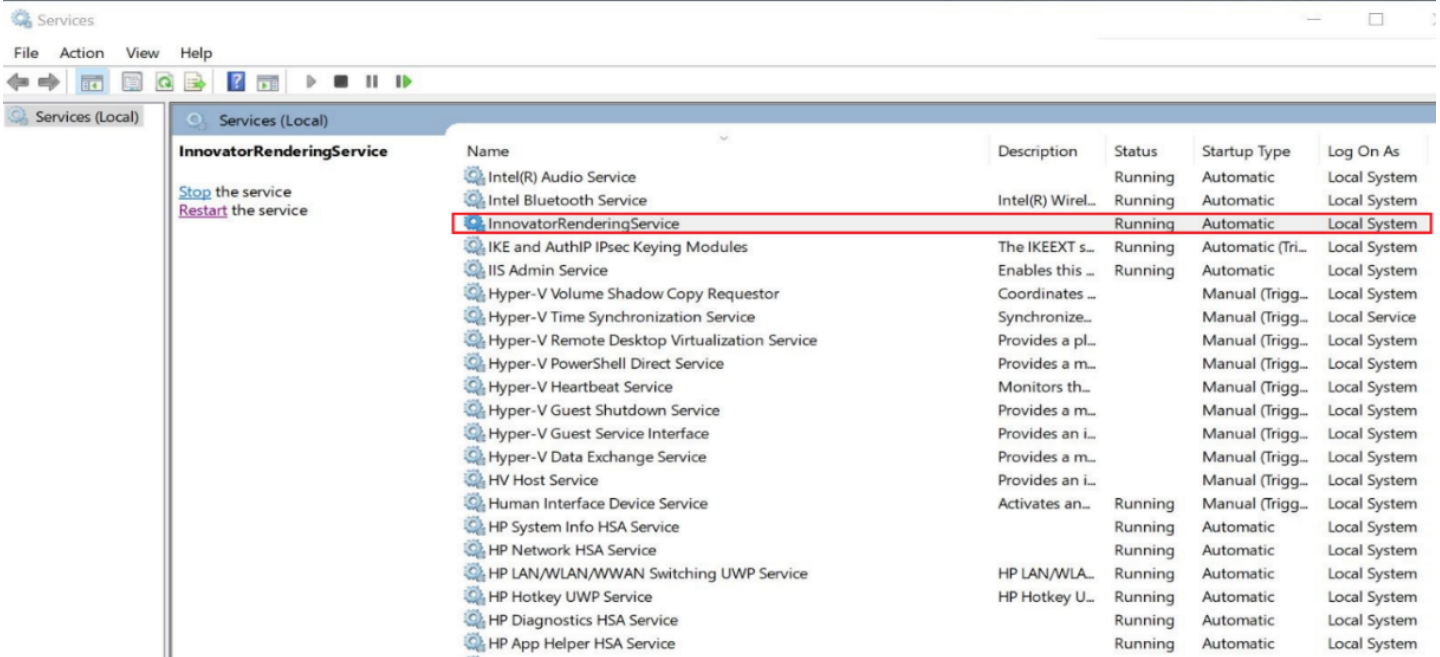
11. In the `Hoops Server` folder, right-click `start_service` file and select **Run as Administrator**.





12. Open Windows **Services** and check if **InnovatorRenderingService** is added and is running.





13. Copy the `ConversionServer` folder from the `CADConverter` package folder (Aras 3D Visualization 33 CD Image\Packages\CADConverter) to a folder that includes the `ConversionServer` folder with the installed Conversion Server.

Replace the files in the destination if the system prompts.

14. Open `ConversionServerConfig.xml` file in a text editor run as administrator. This file is in the root Aras Innovator code tree folder in the case of the default installation.

15. In the `<sectionGroup>` tag, add the following:

```
<section name="ArasCadConverter"
type="Aras.ConversionFramework.Converter.Hoops.Configuration.HoopsConverterConf
ArasCadConverter"> </section>
<section name="ArasCadConverterPrc"
type="Aras.ConversionFramework.Converter.Hoops.Configuration.HoopsConverterConf
ArasCadConverter"> </section>
```

16. In the `< Converters >` tag, add the following:

```
<Converters>
<Converter name="Aras CAD to PDF Converter"
type="Aras.ConversionFramework.Converter.Hoops.HoopsConverter,
ArasCadConverter" /> <Converter name="Aras PRC to SCS Converter"
```



```
type="Aras.ConversionFramework.Converter.Hoops.HoopsConverterPrc,
ArasCadConverter" />
</Converters>
```

17. In the `< ConverterSettings >` tag, add the following:

```
<ConverterSettings>

<!-- Place here configuration sections for converters -->

<ArasCadConverter>

  <Application
converterPath="C:\Aras\14SP10\HOOPSCONVERTER\bin\converter.exe" />

  <Command arguments="--sc compute bounding boxes 'All' --
input pdf template file 'C:\Aras\ Innovator Server Name\HOOPS
Converter\Templates\Blank Template L.pdf' --output pdf
'%filepath%\%filename%.pdf' --output png '%filepath%\%filename%.png' --
output png resolution '150x150' --output sc '%filepath%\%filename%' --
sc create scz 'True' --sc compress scz 'false --output xml assemblytree
'%filepath%\%filename%.xml' --output prc '%filepath%\%filename%.prc' --
background color '1.0, 1.0, 1.0' --output_logfile
'%filepath%\%filename%.log'" />

  <Output>

  <UploadToVault>

  <File extension="prc" argsMarkers="--output_prc" />
  <File extension="scs" argsMarkers="--output_scs" />
  <File extension="pdf" argsMarkers="--output_pdf" />
  <File extension="png" argsMarkers="--output_png" />
  <File extension="stl" argsMarkers="--output_stl" />
```



```
<File extension="xml" argsMarkers="--output_xml_assemblytree" />
```

```
<File extension="scz" argsMarkers="--output_sc" />
```

```
</UploadToVault>
```

```
</Output>
```

```
<AssemblyCommand arguments="--sc compute bounding boxes 'All' --
input pdf template file 'C:\Aras\Innovator Server Name\HOOPS
Converter\Templates\Blank Template L.pdf' --output pdf
'%filepath%\%filename%.pdf' --output_png '%filepath%\%filename%.png' --
output png resolution '150x150' --sc create scz 'true' --sc compress scz
'false' --output xml assemblytree '%filepath%\%filename%.xml' --output_prc
'%filepath%\%filename%.prc' --background color '1.0, 1.0, 1.0' --
output_log file '%filepath%\%filename%.log'" streamingEnabled="True"/>
```

```
</ArasCadConverter>
```

```
<ArasCadConverterPrc>
```

```
<Application converterPath="C:\Aras\ Innovator Server Name \HOOPS
Converter\bin\converter.exe" />
```

```
<Command arguments="--output scs '%filepath%\%filename%.scs' --
output xml assemblytree '%filepath%\%filename%.xml' --output_logfile
'%filepath%\%filename%.log'" />
```

```
<Output>
```

```
<UploadToVault>
```

```
<File extension="prc" argsMarkers="--output_prc" />
```

```
<File extension="scs" argsMarkers="--output_scs" />
```

```
<File extension="pdf" argsMarkers="--output_pdf" />
```



```

<File extension="png" argsMarkers="--output_png" />
<File extension="stl" argsMarkers="--output_stl" />
<File extension="xml" argsMarkers="--output_xml_assemblytree" />
</UploadToVault>
</Output>
</ArasCadConverterPrc>
</ConverterSettings>

```

18. Save and close the `ConversionServerConfig.xml` file.
19. Restart the Internet **Information Services (IIS)**.
20. Copy the `Innovator` folder from the **3DViewers** package folder ( `Aras 3D Visualization 35 CD Image\Packages\3DViewers` ) to the root Aras Innovator code tree folder where the `Innovator` folder exists. Replace the files in the destination if the system prompts.
21. Copy the `Innovator` folder from the **DPN** package folder ( `Aras 3D Visualization 35 CD Image\Packages\DPN` ) to the root Aras Innovator code tree folder where the `Innovator` folder exists. Replace the files in the destination if the system prompts to.
22. In the Aras Innovator code tree, navigate to `\Innovator\Server` and open the `method-config.xml` file in a text editor run as administrator.
23. In the `< ReferencedAssemblies >` tag, add the following:

```

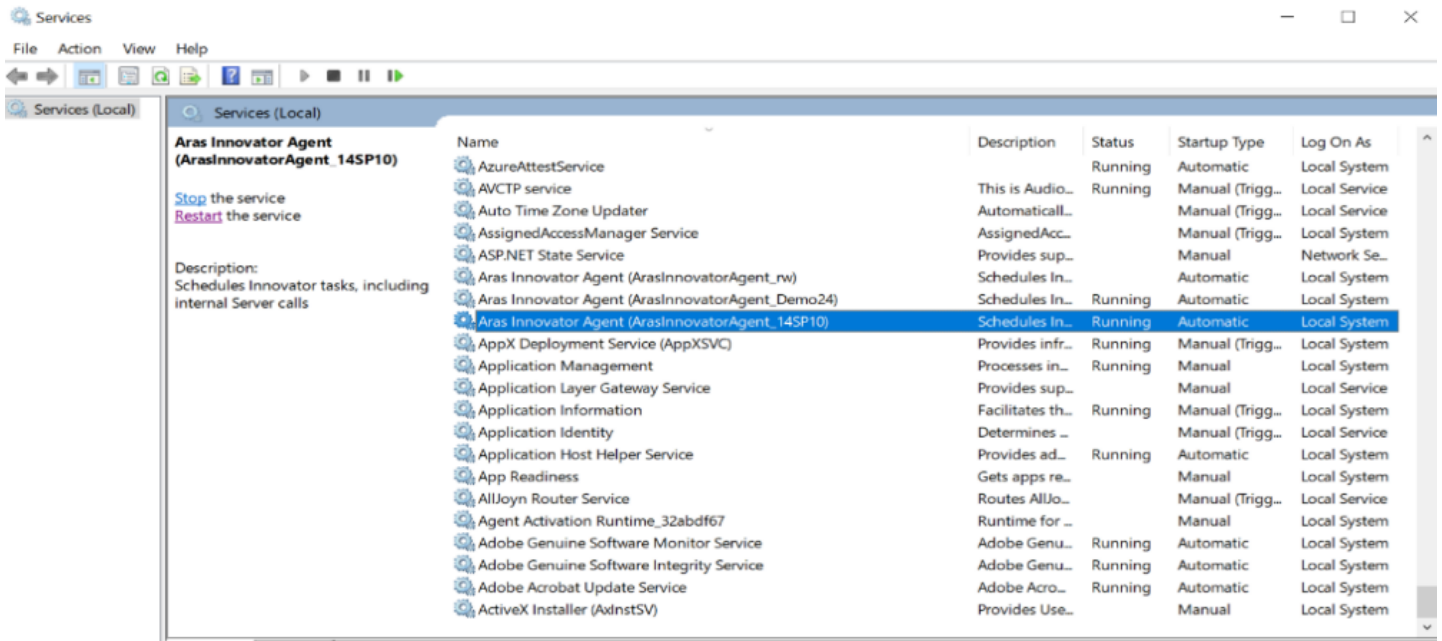
<ReferencedAssemblies>
...
<name>$(binpath)/Aras.DynamicModelViewer.Core.dll</name>
<name>$(binpath)/Aras.DynamicModelViewer.DataModel.dll</name>
<name>$(binpath)/Aras.DynamicModelViewer.QueryProcessor.dll</name>
<name>Microsoft.Extensions.Logging.dll</name>
</ReferencedAssemblies>

```

24. **Save** and close the `method-config.xml` file.



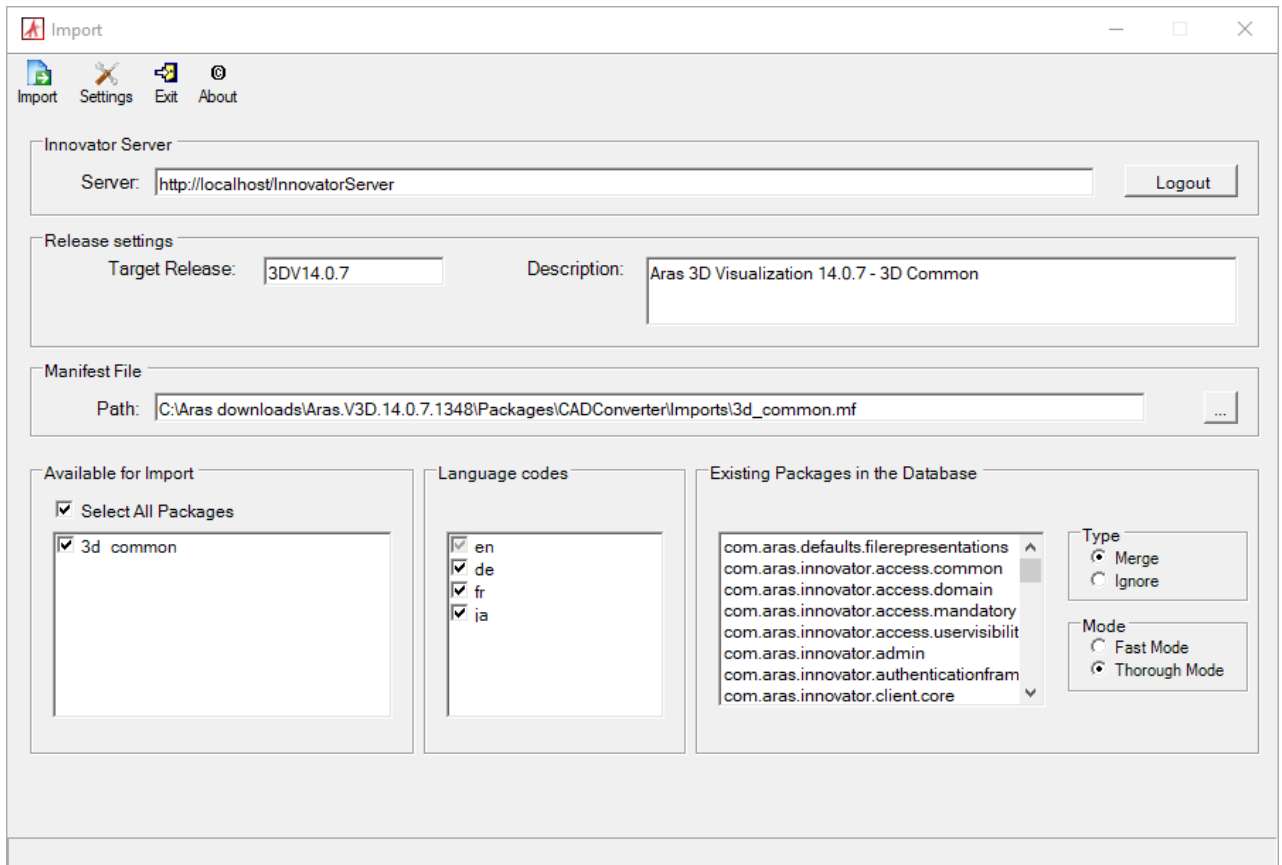
25. Restart the **Internet Information Services (IIS)**.
26. Restart **Aras Innovator Agent** for corresponding instance on Server side on **Windows Services**.



27. Import the Aras 3D Visualization 3D Common database package with the **Package Import Export Utilities**. For more information using this tool, refer to the *Aras Innovator 35 – Package Import Export Utilities* documentation.

1. Enable the **Super User (root)** login.
2. Browse to the `\ PackageImportExportUtilities\Import\ f` folder and run the **Import.exe** file.





3. Input the connection information.

- **Server:** The connection URL for Aras Innovator. By default, it is `http://localhost/InnovatorServer/`. Click the **Login** button and enter:
  - **Database:** The target Aras Innovator database. By default, it is `InnovatorSolutions`.
  - **Username:** `root`
  - **Password:** Password for “root” login (Default is “innovator”)
- **Target Release:** `3DV14.0.8`
- **Description:** `Aras 3D Visualization 14.0.8 – 3D Common`
- **Manifest File Path:** The manifest file `\Packages\CADConverter\Imports\3d_common.mf`
- **Available for Import:** Select All Packages
- **Type:** Merge
- **Mode:** Thorough Mode

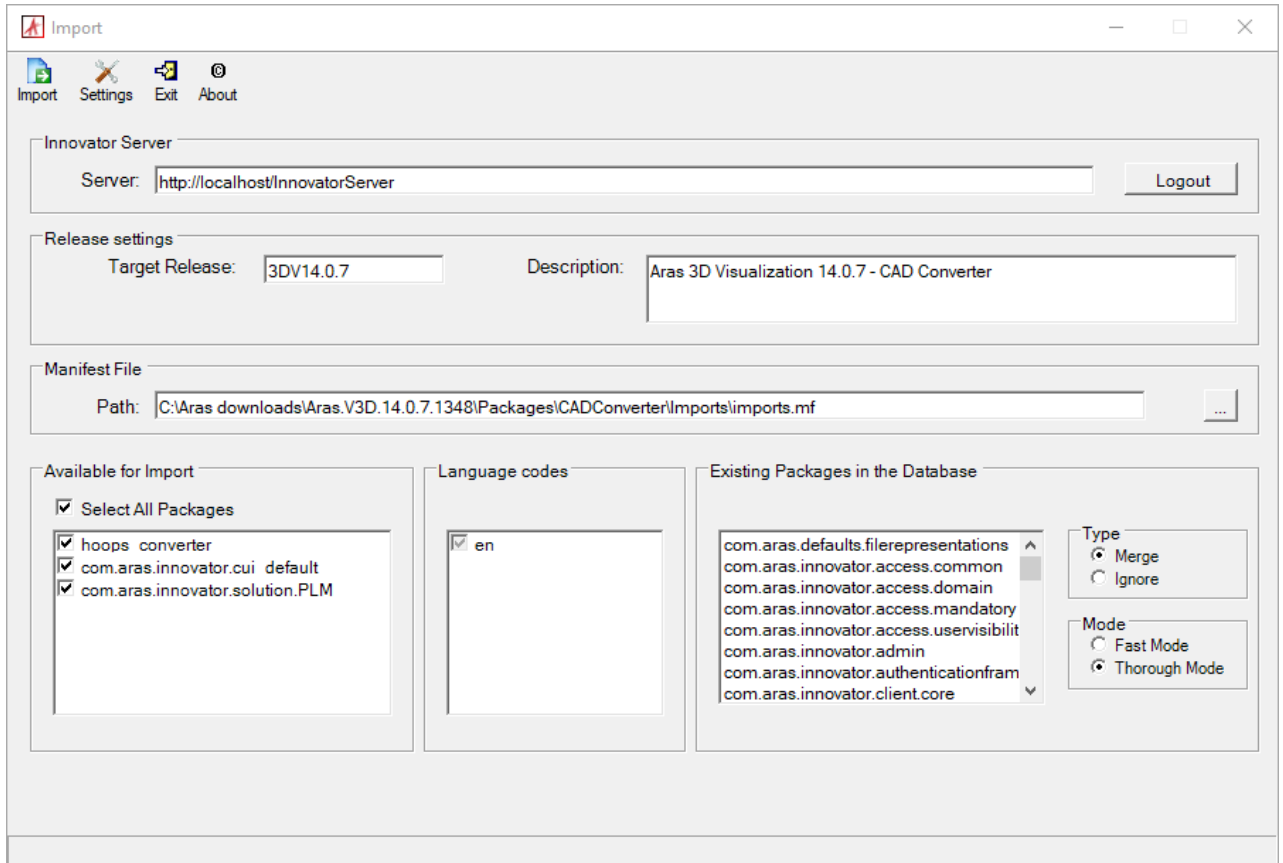
4. Click the **Import** button.

28. Import the Aras 3D Visualization CAD Converter database package with the **Package Import Export Utilities**. For more information using this tool, refer to the *Aras Innovator 35 – Package Import Export Utilities* documentation.

1. Enable the **Super User (root)** login.

2. Browse to the `\PackageImportExportUtilities\Import\ f` folder and run the **Import.exe** file.





3. Input the connection information.

- **Server:** The connection URL for Aras Innovator. By default, it is `http://localhost/InnovatorServer/`. Click the **Login** button and enter:
  - **Database:** The target Aras Innovator database. By default, it is `InnovatorSolutions`.
  - **Username:** `root`
  - **Password:** Password for “root” login (Default is “innovator”)
- **Target Release:** `3DV14.0.8`
- **Description:** `Aras 3D Visualization 14.0.8 – CAD Converter`
- **Manifest File Path:** The manifest file `\Packages\CADConverter\Imports\imports.mf`
- **Available for Import:** `Select All Packages`
- **Type:** `Merge`
- **Mode:** `Thorough Mode`

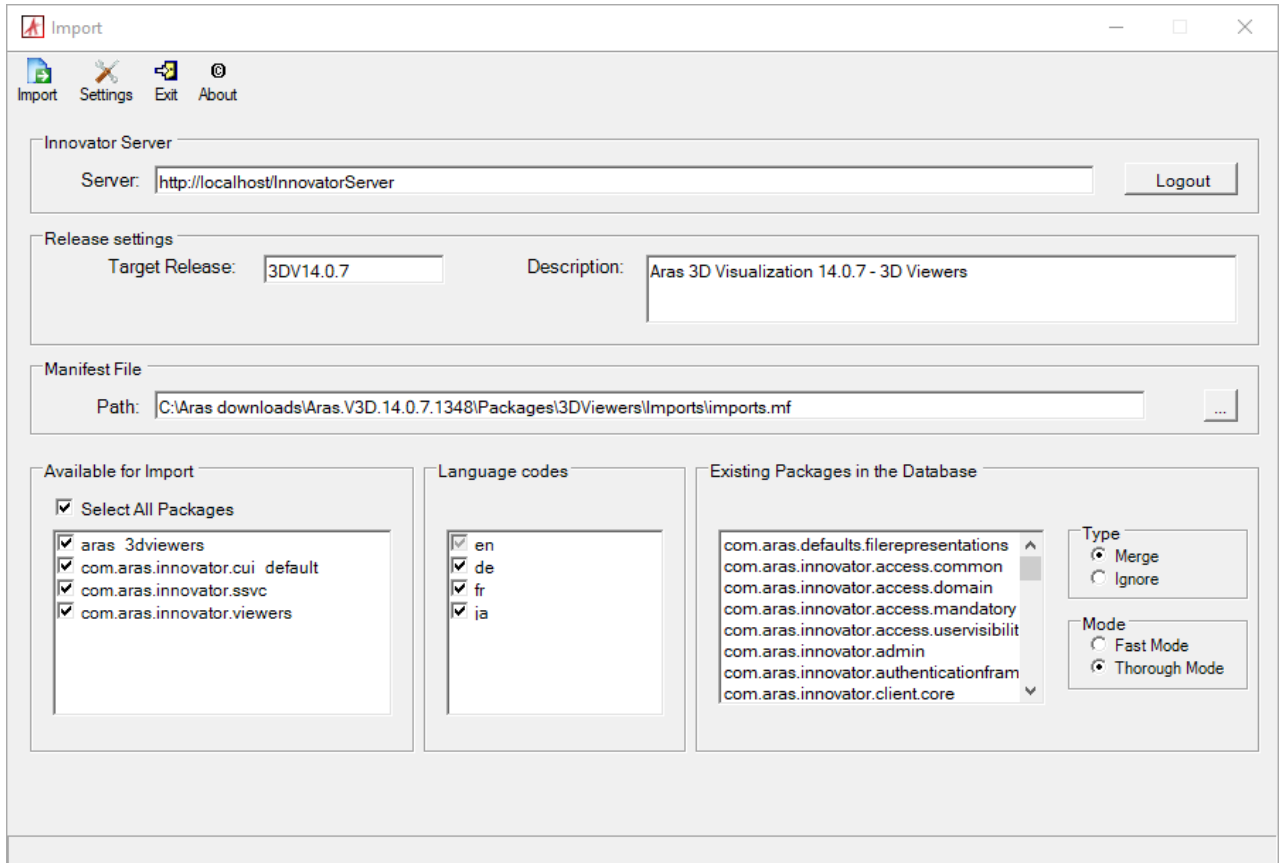
4. Click the **Import** button.

29. Import the Aras 3D Visualization 3D Viewers database package with the **Package Import Export Utilities**. For more information using this tool, refer to the *Aras Innovator 35 – Package Import Export Utilities* documentation.

1. Enable the **Super User (root)** login.

2. Browse to the `\PackageImportExportUtilities\Import\` folder and run the **Import.exe** file.





3. Input the connection information.

- **Server:** The connection URL for Aras Innovator. By default, it is `http://localhost/InnovatorServer/`.
- Click the **Login** button and enter:
  - **Database:** The target Aras Innovator database. By default, it is `InnovatorSolutions`.
  - **Username:** `root`
  - **Password:** Password for “root” login (Default is “innovator”)
- **Target Release:** `3DV14.0.8`
- **Description:** `Aras 3D Visualization 14.0.8 – 3D Viewers`
- **Manifest File Path:** The manifest file `\Packages\3DViewers\Imports\imports.mf`
- **Available for Import:** `Select All Packages`
- **Type:** `Merge`
- **Mode:** `Thorough Mode`

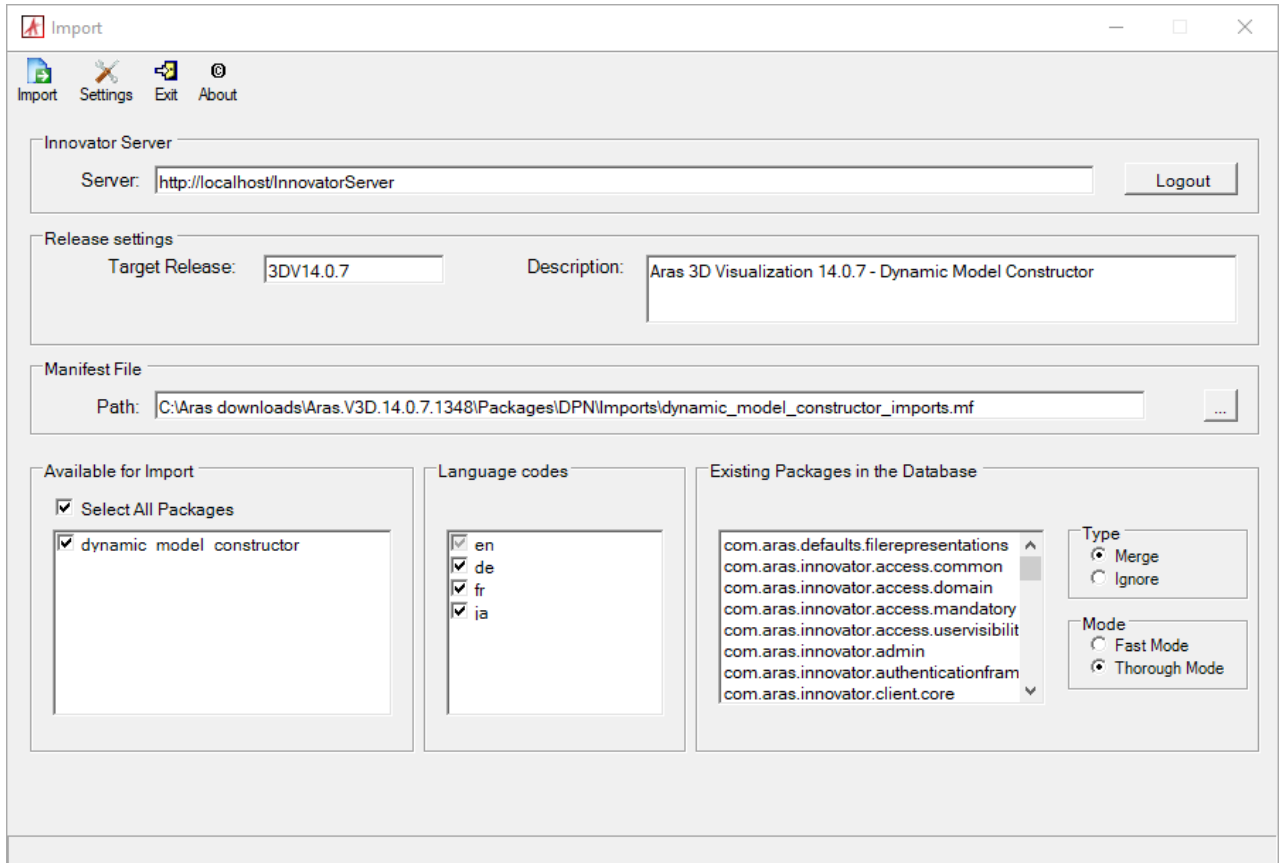
4. Click the **Import** button.

30. Import the Aras 3D Visualization Dynamic Model Constructor database package with the **Package Import Export Utilities**. For more information using this tool, refer to the *Aras Innovator 35 – Package Import Export Utilities* documentation.

1. Enable the **Super User (root)** login.

2. Browse to the `\PackageImportExportUtilities\Import\` folder and run the **Import.exe** file.





3. Input the connection information.

- **Server:** The connection URL for Aras Innovator. By default, it is `http://localhost/InnovatorServer/`.
- Click the **Login** button and enter:
  - **Database:** The target Aras Innovator database. By default, it is `InnovatorSolutions`.
  - **Username:** `root`
  - **Password:** Password for “root” login (Default is “innovator”)
- **Target Release:** `3DV14.0.8`
- **Description:** `Aras 3D Visualization 14.0.8 – Dynamic Model Constructor`
- **Manifest File Path:** The manifest file  
`\Packages\DPN\Imports\dynamic_model_constructor_imports.mf`
- **Available for Import:** Select All Packages
- **Type:** Merge
- **Mode:** Thorough Mode

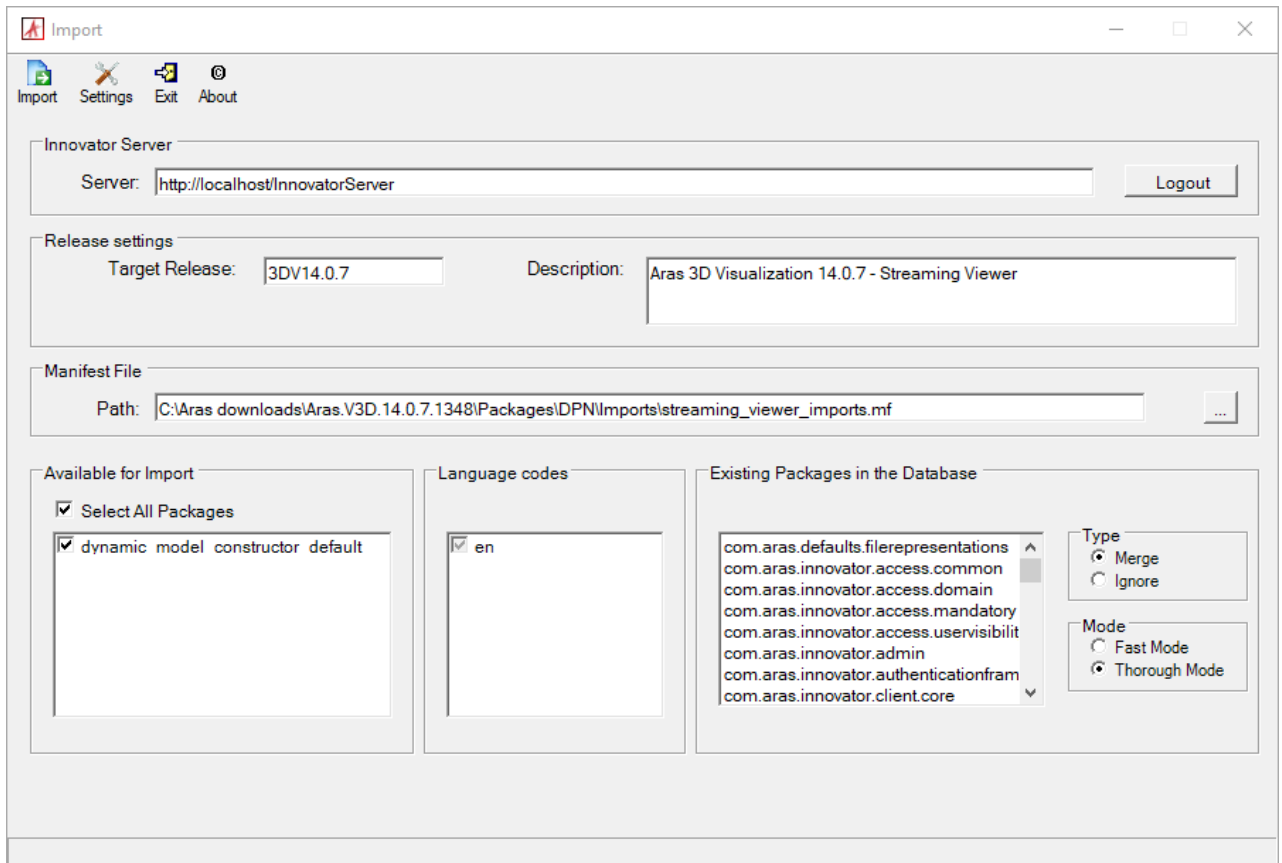
4. Click the **Import** button.

31. Import the Aras 3D Visualization Streaming Viewer database package with the **Package Import Export Utilities**. For more information using this tool, refer to the *Aras Innovator 35 – Package Import Export Utilities* documentation.

1. Enable the **Super User (root)** login.



- Browse to the `\PackageImportExportUtilities\Import\` folder and run the **Import.exe** file.



- Input the connection information.
  - Server:** The connection URL for Aras Innovator. By default, it is `http://localhost/InnovatorServer/`. Click the **Login** button and enter:
    - Database:** The target Aras Innovator database. By default, it is `InnovatorSolutions`.
    - Username:** `root`
    - Password:** Password for “root” login (Default is “innovator”)
  - Target Release:** `3DV14.0.8`
  - Description:** `Aras 3D Visualization 14.0.8 – Streaming Viewer`
  - Manifest File Path:** The manifest file `\Packages\DPN\Imports\streaming_viewer_imports.mf`
  - Available for Import:** Select All Packages
  - Type:** Merge
  - Mode:** Thorough Mode
- Click the **Import** button.

- Disable the **Super User (root)** login.



### Important

The **Super User (root)** login should not be enabled in production.

33. This step is optional. Confirm the successful installation. See the *Confirming Aras 3DV Installation* section.
34. For the required setup to get the streaming viewer running, refer to the *Administrative Configuration for Streaming Viewer* section.
35. If the installation fails, restore the Aras Innovator code tree, Conversion Server, and database with the backups done before the installation and contact Aras Support at [support@aras.com](mailto:support@aras.com) .



## Confirming Aras 3DV Installation

The following steps outline the process of confirming the successful installation of given Aras 3D Visualization components:

1. Log into Aras Innovator as an administrator.
2. From the **Table of Contents**, expand **Administration** and select **Variables**.
3. Confirm that Variables for the given components are listed with necessary Values:
  - o 3D Visualization.3DViewers: 14.0.7
  - o 3D Visualization.CadConverter: 14.0.7
  - o 3D Visualization.DPN: 14.0.7
  - o 3D Visualization.restoreCamera: true

Name ↑	Value	Default Value
3D Visualization.3DViewers	14.0.7	
3D Visualization.CadConverter	14.0.7	
3D Visualization.DPN	14.0.7	
3D Visualization.RestoreCamera	true	true

If there is no Variable with the correct Value for a component, the installation of this component fails. Restore the Aras Innovator code tree, Conversion Server, and database with the backups done before the installation and contact Aras Support at [support@aras.com](mailto:support@aras.com) .



## Troubleshooting

### [Conversion Tasks Not Created](#)

If more than one **FileType** Item has the same file extension set, newly added files automatically choose the **FileType** Item that comes first alphabetically.

Due to this behavior, Conversion Rules should specify all possible **FileType** Items for each file extension that should be included in the Rules.



## Unauthorized Access Exception

### Error Message

```
System.Web.Services.Protocols.SoapException:  
System.Web.Services.Protocols.SoapException: Server was unable to process  
request. ---> System.UnauthorizedAccessException: Access to the path '...  
\ConversionServer\temp\...' is denied.  
  
at System.IO.__Error.WinIOError(Int32 errorCode, String maybeFullPath)  
  
at System.IO.Directory.InternalCreateDirectory(String fullPath, String  
path, DirectorySecurity dirSecurity)  
  
at System.IO.Directory.CreateDirectory(String path, DirectorySecurity  
directorySecurity)  
  
...
```

### Solution

The following steps outline the solution to the Unauthorized Access Exception:

1. Navigate to the `ConversionServer` folder following the path given in the error message.
2. Right-click the `ConversionServer` folder and click **Properties**.
3. On the **Security** tab, click **Advanced**.
4. Click **Change Permissions**.
5. Click **Add**.
6. Type **Everyone** and click **OK**.
7. Select the **Full control** checkbox for **Access** and **Allow** for **Type**.
8. Click **OK**.
9. Select **Replace all child object permissions with inheritable permissions from this object**.
10. Click **Apply** and click **Yes** when prompted.



## 500.19 Internal Server Error

### Error Message

System.InvalidOperationException: The client found the response content type of 'text/html; charset=utf-8', but expected 'text/xml'.

The request failed with the following error message:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>IIS 8.0 Detailed Error - 500.19 - Internal Server Error</title>
...
<fieldset><legend>Error Summary</legend>
<h2>HTTP Error 500.19 - Internal Server Error</h2>
<h3>The requested page cannot be accessed because the related
configuration data for the page is invalid.</h3>
...
```

### Solution

Check for capitalization differences, typos, extra newlines, and carriage returns in the `ConversionServerConfig.xml` file located in the `ConversionServer` folder with the installed Conversion Server. This file is modified when installing the Aras CAD Converter manually; see the Manual Aras CAD Converter Installation section. An entire `<... arguments="..." .../>` attribute must be only one single line without carriage returns or newlines.



## 404: Not Found

### Error Message

```
System.Net.WebException: The request failed with HTTP status 404: Not Found.
```

```
at  
System.Web.Services.Protocols.SoapHttpClientProtocol.ReadResponse(SoapClientMessage message, WebResponse response, Stream responseStream, Boolean asyncCall)
```

```
at  
System.Web.Services.Protocols.SoapHttpClientProtocol.EndInvoke(IAsyncResult asyncResult)
```

```
at  
Aras.ConversionFramework.Proxy.ConversionService.EndConvert(IAsyncResult asyncResult) in d:\Builds\Daily\5779\RELS9-3\Innovator\CompilableCode\_CommonFiles\ConversionService\ConversionService.vb:line 78
```

```
at Aras.Server.Core.InvokeStandardConverter.ConvertCallback(IAsyncResult ar) in d:\Builds\Daily\5779\RELS9-3\Innovator\Server\src\Core\InternalMethods\InvokeStandardConverter.vb:line 85
```

```
at System.Web.Services.Protocols.WebClientAsyncResult.Complete()
```

```
at  
System.Web.Services.Protocols.WebClientProtocol.ProcessAsyncResponseStreamResult(client, IAsyncResult asyncResult)
```

```
at  
System.Web.Services.Protocols.WebClientProtocol.ReadAsyncResponseStream(WebClient client) at
```



```
System.Web.Services.Protocols.WebClientProtocol.ReadAsyncResponse(WebClientAsync  
client)
```

## Solution

Confirm that the Conversion Server URL is correct in Aras Innovator.

This URL depends on where the Conversion Server was installed on the website with the Aras Innovator installer. For example, the default URL is the following:

- When the Conversion Server was installed with the Aras Innovator instance:  
<http://serverName/InnovatorServer/ConversionServer/ConversionService.aspx>
- When only the Conversion Server was installed: <http://serverName/ConversionServer/ConversionService.aspx>

where, serverName is the Aras Innovator domain name.

## 3D PDF Is Not Activated in Firefox

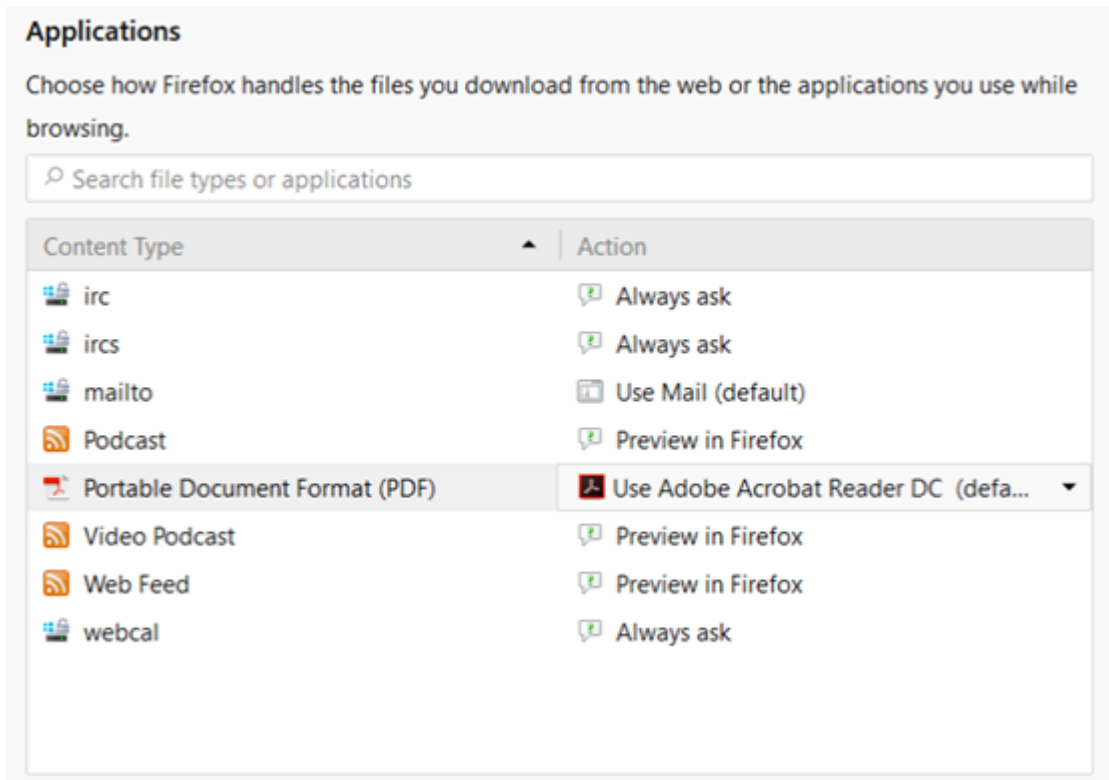
### Error

Clicking a CAD image in a converted 3D PDF file activates the ability to manipulate the CAD content interactively. If this feature is not available in a 3D PDF file opened in Firefox, this file may be loaded without using Adobe Reader.

### Solution

The following process outlines the solution to configure Firefox to load PDF files using Adobe Reader:

1. Open Mozilla Firefox.
2. Go to **Settings** and select **General**
3. Click **Applications**.
4. Set the **Portable Document Format (PDF)** Content Type to the **Use Adobe Acrobat Reader (default)** Action.



# Upgrading from Version 14+



## Introduction

### Purpose

This Upgrade Guide describes how to upgrade from Aras 3D Visualization (3DV) 33 to 35.



## Scope

This Upgrade Guide provides detailed information and instructions for upgrading the Aras 3D Visualization (3DV) platform component from version 19, 21, 26, 29, 31, 33 to 35.



## Target Audience

This document is intended for administrators responsible for upgrading the Aras 3D Visualization platform component. Experience with upgrading Aras Innovator is required by administrators installing the Aras 3DV platform component.



## Prerequisites

To upgrade the Aras 3D Visualization platform component to version 35, earlier versions such as versions 19, 21, 26, 29, 31 or 33 should be installed. If upgrading from a version earlier than 14, please contact Aras Support at [support@aras.com](mailto:support@aras.com).

The following prerequisites are required to upgrade to Aras 3D Visualization 35:

- Aras 3D Visualization Release 14, 21, 26, 29, 31, or 33
- Aras Innovator Release 20 – 35
- Aras Update 1.23+
- Feature License Key: **Aras.ApplyUpdate**

### Important

The upgrade process includes applying encrypted database patches, which requires using Aras Update tool. As a result, the manual upgrade option cannot be offered for Aras 3D Visualization 35, and only the automated option using Aras Update is therefore available.

The upgrade scripts for Aras Innovator require an *Aras.ApplyUpdate* Feature to run. This Feature is available to subscribers as part of the *Aras.EnterpriseSubscription* package.

To check whether a valid Feature License exists in the Aras Innovator Database:

1. Log into Aras Innovator as an administrator.
2. Go to **Administration\Feature Licenses**.
3. Check that *Aras.EnterpriseSubscription* Feature License is present. If this Feature License is found, check the expiration date to confirm that the license is still active.

## Requesting a Feature License

In order to request a feature license activation key, customers with an active Aras subscription can send an email to [licenses@aras.com](mailto:licenses@aras.com) or to their account representative using the following format:

- Subject – “Aras Enterprise Subscription Key Required”
- Body - Version of Aras Innovator

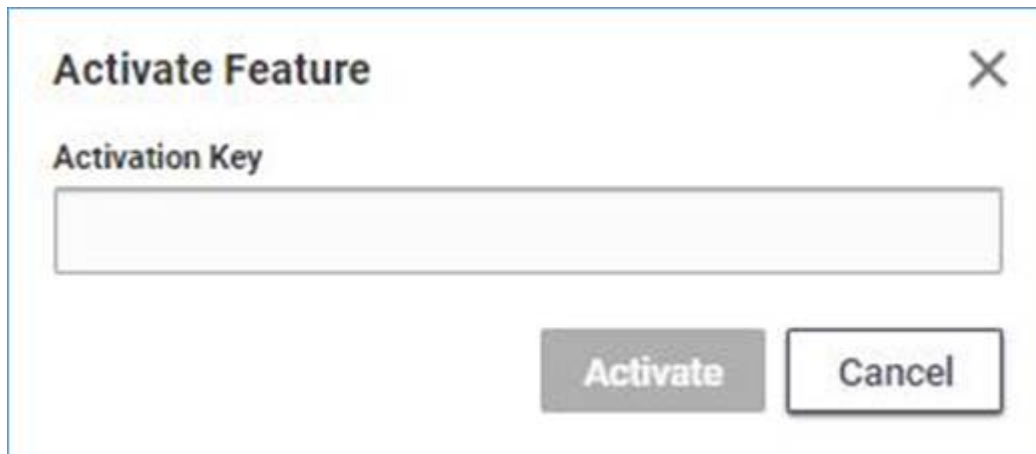
You will receive a reply containing the Feature License activation key.



## Installing the License

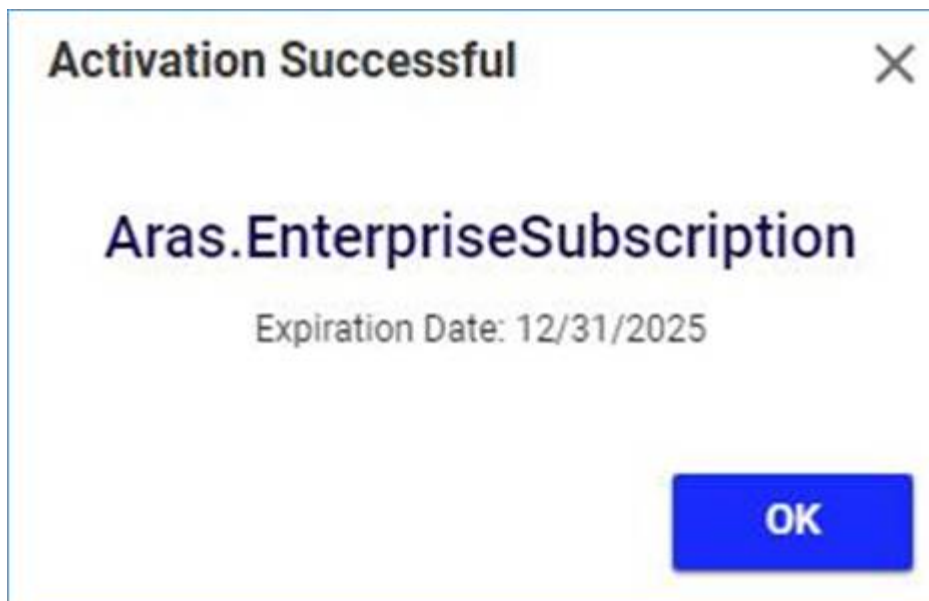
Once you have the activation key, install the feature license as follows:

1. Log into Aras Innovator as an administrator.
2. From the main menu toolbar, select Tools → Admin → Licenses → Activate Feature.



The screenshot shows a dialog box titled "Activate Feature" with a close button (X) in the top right corner. Below the title is a label "Activation Key" followed by a large, empty text input field. At the bottom of the dialog, there are two buttons: "Activate" and "Cancel".

3. Paste the acquired activation key and click Activate Feature button.  
A success message similar to the following should appear.



The screenshot shows a dialog box titled "Activation Successful" with a close button (X) in the top right corner. The main text in the center reads "Aras.EnterpriseSubscription" in a large, bold font. Below this, it says "Expiration Date: 12/31/2025". At the bottom right of the dialog, there is a blue button with the text "OK".

4. Go to the **Innovator Admin menu** → **Licenses** → **Update Feature Tree**  
A success message similar to the following should appear.



## Eligibility Verification

The following steps outline the process to verify the current Aras Innovator code tree and database meet the minimum requirements to apply the Aras Component Engineering updates:

1. Log into **Aras Innovator** as an **Administrator**.
2. From the **Main Menu**, select Help and About.
3. Verify the dialog displays a release number between **Release 20** and **Release 35**
4. Close the dialog.
5. From the **TOC**, select **Administration** and **Variables**.
6. Confirm the following variables display:



## Aras 3D Visualization Download

The Aras 3D Visualization 35 platform component is delivered as the **Aras 3D Visualization 35 CD Image** package, which subscribers can download from the [Aras File Sharing](#) site. The site content and folder structure are specific for a given logged-in user. The downloaded package may be a zip archive that should be extracted. The unzipped package should be copied to a machine where an Aras 3D Visualization component is going to be upgraded.



## Upgrade Overview

Aras 3D Visualization consists of multiple components that can be upgraded with the standard Aras tools following different scenarios.

As the Aras 3DV upgrade modifies the Aras Innovator code tree, Conversion Server, and database, they must be backed up before the upgrade for their revert in an upgrade failure case.

Aras Innovator will be down during the upgrade.

It is always recommended to test the Aras 3DV upgrade procedure on a non-production Aras Innovator instance before running this procedure on a production system.

If the Dynamic 3D Viewer has not been installed with an older version, it can be installed with the new version during the upgrade.

The Streaming 3D Viewer can be installed during the upgrade. See *Automated Aras 3DV Installation for Streaming Viewer* section in the *Aras 3D Visualization 35 - Installation Guide* for more information.

## Upgrade Tools

The Aras 3D Visualization can be only upgraded with:

- Aras Update 1.23 or higher.

The upgrade process is automated.

It is possible to upgrade all components installed on one machine at once. If the components are installed on multiple machines, the upgrade procedure should be performed for every machine in the order discussed in the *Upgrade Scenarios* section.

For the procedure, see section *Automated Upgrade*.

This tool requires installation. By default, the **ArasUpdate.exe** file is installed in the following directory: **C:\Program Files (x86)\Aras\Aras Update**. For tool details, see the *Aras Update – Installation Guide* and *Aras Update – User Guide*.



## Aras 3D Visualization Components and Modules

Aras 3D Visualization includes the following components:

- **Aras CAD Converter:** provides the ability to convert native 3D CAD files for use with 3D Viewers. This component also includes the HOOPS Converter and updates to the Conversion Server.
- **Aras 3D Viewers:** provides core 3D Visualization functionality for the Monolithic and the Dynamic 3D Viewers.
- **Aras Dynamic Visualization:** provides the ability to configure 3D and Tree Grid Views for the Dynamic 3D Viewer based on selected Query and Dynamic View Definitions. This component is optional and requires both other components.
- **Aras Streaming Viewer (Hoops Server):** The Hoops Server provides the ability to use HOOPS Communicator server-side streaming for geometry data.

One, many, or all Aras 3DV components can be installed on one or multiple machines.

Aras CAD Converter, Aras 3D Viewer and Aras Dynamic Visualization components are delivered as two installation modules with updates for:

- Code tree
- Database

Thus, a component code tree and database can be installed on one or multiple machines. Additionally, the component code tree can be installed on one or multiple machines. The Aras Dynamic Visualization code tree should be on a single machine.

## Upgrade Scenarios

To upgrade a component, its code tree and database should be both updated; otherwise, the component will not be successfully upgraded. It is possible to upgrade the component code tree and database in separate sessions in an arbitrary order code tree and then database or vice versa.

All installed components of an earlier version must be updated for a successful Aras 3D Visualization upgrade in the following order:

1. Aras CAD Converter
2. Aras 3D Viewers
3. Aras Dynamic Visualization



## Installing Dynamic 3D Viewer During Upgrade

If installing the Dynamic 3D Viewer (Aras Dynamic Visualization) for the first time during the upgrade, the Dynamic Enable process should be activated for the CAD assemblies that were viewed with the Monolithic 3D Viewer because Aras CAD Converter has not generated data for viewing them in the Dynamic 3D Viewer.

For the process details and activation procedure, see the *Dynamic Enabling* section in the *Aras 3D Visualization 35 - Administrator Guide*.



## Installing Streaming 3D Viewer During Upgrade

### Important

Please note that the output files of Dynamic/Monolithic Viewer and Streaming Viewer are incompatible with one another.

If the Streaming 3D Viewer is installed in an environment where the Monolithic or Dynamic 3D Viewers have been previously installed and used, all existing native files of existing CAD Documents need to be re-converted. There is no automated means to perform this reversion.

### Important

Only one Streaming Viewer can be installed on one machine at a time.

### Important

The Streaming Viewer currently cannot be deployed in a cloud environment.

The HOOPS Server must be deployed with networked file access to a single vault containing view files for rendering.



## Pre-Upgrade Procedures

Before upgrading Aras 3D Visualization:

- Users should be informed that Aras Innovator will be down.
- The Aras Innovator code tree, Conversion Server, HOOPS Converter, and database should be backed up. Such files should be placed in a safe location as they will be used for reverting to the pre-upgrade condition if the upgrade fails.
- If installed, the Streaming Service must be shut down prior to upgrading. As an Administrator, open Windows **Services** and find **InnovatorRenderingService**. Right-Click on it and **stop** the service.

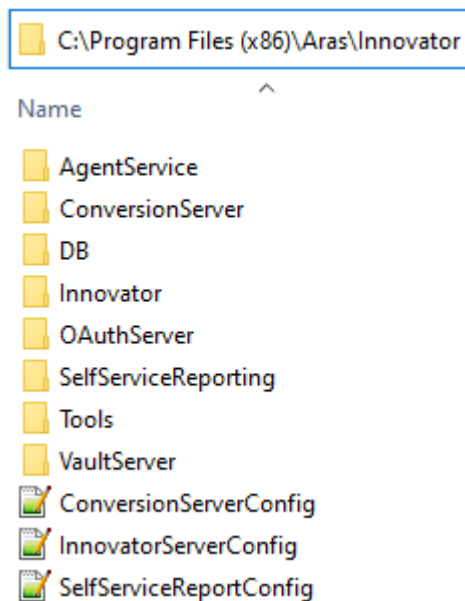


## Aras Innovator Code Tree

The Aras Innovator code tree is inclusive of files and folders installed on a disk with the first Aras Innovator installation.

The default code tree installation path is the following: C:\Program Files(x86)\Aras\Innovator

The following figure shows the out-of-the-box code tree content.



The following steps outline the process to locate the installation path, if Aras Innovator was installed in a different location.

1. Click the Windows **Start** button and go to **Control Panel**.
2. Select **Programs and Features**.
3. Click **Aras Innovator** from the installed applications list. The full installation path appears in the **Comments** field.

Programs and Features

Control Panel > Programs > Programs and Features

Control Panel Home

Uninstall or change a program

To uninstall a program, select it from the list and then click Uninstall, Change, or Repair.

View installed updates

Turn Windows features on or off

Install a program from the network

Name	Publisher	Installed On	Size	Version
Aras Innovator	Aras Corporation	6/15/2022	943 MB	14.0.7
Aras Office Connector	Aras Corporation	2/9/2022	11.8 MB	12.0.5.1957
Aras Update	Aras Corporation	5/6/2022	7.14 MB	1.17.1264

Aras Corporation Product version: 14.0.7 Support link: <http://www.ArasCorp...> Help link: <http://www.ArasCorp...> Size: 943 MB

Comments: Installed to C:\Program Files (x86)\Aras\Innovator\



## Notification and Backup

The following steps are to be followed before upgrading Aras 3D Visualization:

1. Notify users that:
  - o Aras Innovator will be down at a scheduled time.
  - o They should log out of Aras Innovator prior to starting the installation.

It is best to give at least a 24-hour notice and a reminder 15 minutes before taking the platform offline.

2. Back up the Aras Innovator code tree.
3. Back up the Conversion Server.  
Its default installation path is the following: **C:\Program Files(x86)\Aras\ConversionServer**
4. Back up the HOOPS Converter.  
Its default installation path is the following: **C:\HOOPS Converter**
5. Disconnect all users from the database to prevent client sessions from committing any further changes to the database.  
The easiest way is to change the database connection string from `<DB-Connection...` to `<xDB-Connection` in the `InnovatorServerConfig.xml` file and restart the **w3svc** service (IIS). This expires all sessions and prevents all new connections to the Aras Innovator database through the existing instance.
6. Back up the database.
7. After the database backup has been completed, enable the database connections.  
If the abovementioned way is followed, change the database connection string from `<xDB-Connection...` to `<DB-Connection` in the **InnovatorServerConfig.xml** file and restart the **w3svc** service (IIS).



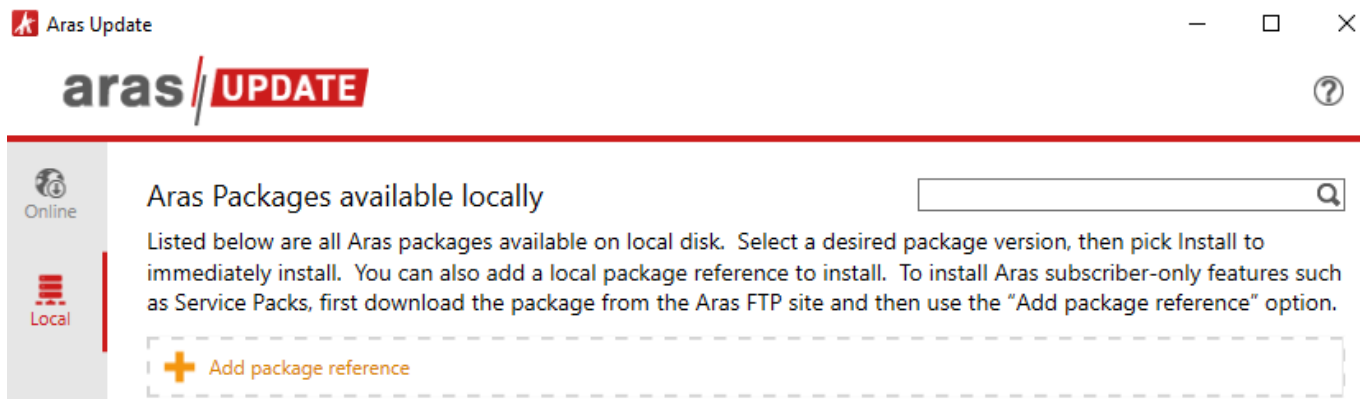
## Automated Upgrade

Aras Update provides an automated application upgrade process.

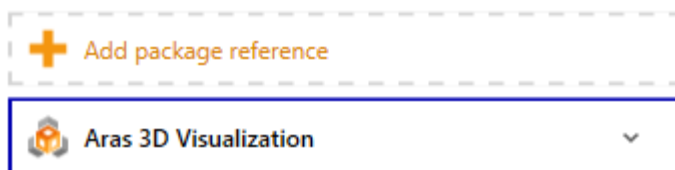
The procedure is the same when upgrading one, multiple, or all Aras 3D Visualization components on one machine. If upgrading one, multiple, or all components on multiple machines, the procedure should be performed on each machine in the order discussed in section Upgrade Scenarios.

To upgrade one, multiple, or all Aras 3DV components on one machine using Aras Update:

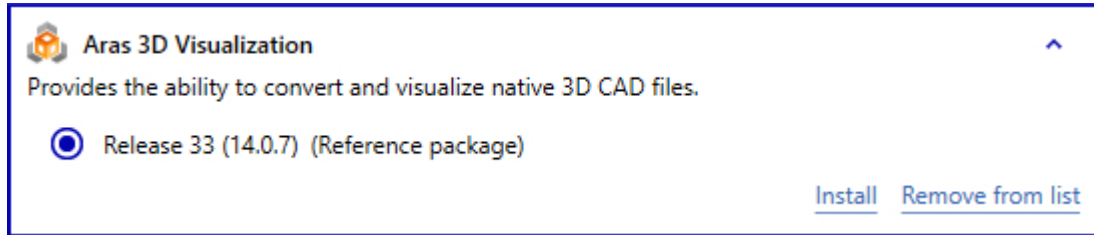
1. Perform the pre-upgrade procedures as discussed in the section Pre-Upgrade Procedures.
2. Copy the unzipped Aras 3D Visualization 33 CD Image package to a target machine.
3. Enable logon for the root (Super User) User.
4. Launch the Aras Update tool as an administrator.
5. In Aras Update, go to the Local sidebar tab and click + Add package reference. A search dialog appears.



6. Search for and select the **Aras 3D Visualization 35 CD Image** folder. The **Aras 3D Visualization** option appears in Aras Update.



7. Click the Aras 3D Visualization Release 35 (14.0.8) option and then click Install.



8. Select the installation modules of the Aras 3D Visualization components that should be upgraded on the given machine and click **Next**.

To install Streaming Viewer during the upgrade, select the **HOOPS Server** checkbox. For more information on installing Streaming Viewer see Automated Aras 3DV Installation for Streaming Viewer section in the in the *Aras 3D Visualization 35 - Installation Guide*.

#### Important

The Aras CAD Converter, Aras 3D Viewer, and Aras Dynamic Visualization component consist of two installation modules: code tree and database updates. To install a component, both of its modules should be installed; otherwise, the component will not be completely installed. It is possible to update the code tree and database in separate installation sessions.



Aras 3D Visualization Release 33 (14.0.7) requires the following components for proper installation. Select the components you intend to install.

Computer name: ARAS-CORP/AC5CG3221VKB

- Aras CAD Converter**  
Provides the ability to convert native 3D CAD files for use with 3D Viewers.
- Aras CAD Converter Database Updates**  
Import to Innovator Database.
- Aras 3D Viewers**  
Provides core 3D Visualization functionality for Monolithic and Dynamic 3D Viewers.
- Aras 3D Viewers Database Updates**  
Import to Innovator Database.
- Aras Dynamic Visualization**  
Provides the ability to configure 3D and Tree Grid Views for the Dynamic 3D Viewer based on selected Query and Dynamic View Definitions.
- Aras Dynamic Visualization Database Updates**  
Import to Innovator Database.
- Hoops Server**  
Provides the ability to use HOOPS Communicator server-side streaming for geometry data.

Back

Next

Cancel

9. Select the appropriate logging option and click **Next**. When selected, a log file records the installation process to troubleshoot issues.



Select if you want to create a log file to record installation transactions.

Log file location: C:\ProgramData\Aras\ArasUpdate\V3D\14.0.5.1075\Logs

Create log file including:

Simple logging

Detailed logging

Back

Next

Cancel

10. Specify the connection information for the installation modules of the components to be upgraded:

Aras Update

# aras UPDATE

Enter the following parameters for each component you are going to install and then click Install button to continue.

- Aras CAD Converter**

Path to root folder where Innovator Server is installed

Specify local directory where HOOPS Converter should be installed (example: 'C:\HOOPS Converter')

Use streaming SCZ by default

- Aras CAD Converter Database Updates**

Server URL (example - 'http://localhost/InnovatorServer')

Database Name (example - 'InnovatorSolutions')

Username (should use 'root' User)

Password for 'root' User

- Aras 3D Viewers**
- Aras 3D Viewers Database Updates**
- Aras Dynamic Visualization**
- Aras Dynamic Visualization Database Updates**
- Hoops Server**



Path to the folder where HOOPS Server should be installed

Hostname of the machine where HOOPS Server should be installed

Path to the vault folder for corresponding Innovator instance

### Component CAD Converter Updates:

- **Path to root folder where Innovator Server is installed:** a physical path to the Aras Innovator code tree. See section *Aras Innovator Code Tree*.
- **Specify local directory where HOOPS Converter should be installed:** a physical path to a local folder where HOOPS Converter files will be installed; for example, **C:\HOOPS Converter**.

#### Important

The **Use streaming SCZ by default** field is selected during the installation of the Streaming Viewer. For more information on installing Streaming Viewer see ***Automated Aras 3DV Installation for Streaming Viewer*** section in the in the Aras 3D Visualization 35 - Installation Guide.

#### Important

If the **Use streaming SCZ by default** checkbox is checked, and the Streaming Service (HOOPS Server) is not installed, none of the other Viewers will work.

### Component Database Updates:

- **Server URL:** a URL for connecting to a given Aras Innovator instance. By default, it is <http://localhost/InnovatorServer/> .
- **Database Name:** a name of a target Aras Innovator database. By default, it is **InnovatorSolutions**.
- Username: root.
- **Password for 'root' User:** a password for the root User. By default, it is **innovator**.



11. If installing multiple Aras 3D Visualization components, connection settings for the following components are automatically inherited from the preceding ones. For example, if Aras 3D Viewers is installed with Aras CAD Converter, the Aras 3D Viewers modules inherit connection settings from the Aras CAD Converter modules.

If upgrading multiple Aras 3DV components during one session, connection settings for the following components are automatically inherited from the preceding components. For example, if Aras 3D Viewers are upgraded with Aras CAD Converter, the Aras 3D Viewers modules inherit connection settings from the Aras CAD Converter modules.

12. Once the connection information is provided, click **Install**. The upgrade process begins.



Installation is in progress. Please, wait...



Installing...

[Show log file](#)

Cancel

13. Once the upgrading is successfully completed, close Aras Update.





Installation completed successfully!

[Show log file](#)

Home

Exit

14. Disable the logon for the root (Super User) User.

15. This step is optional. Confirm the successful Aras 3DV upgrade. See section *Confirming Successful Upgrade*.

If the upgrade fails, restore the Aras Innovator code tree, Conversion Server, HOOPS Converter, and database with the backups done before the upgrade and contact Aras Support at [support@aras.com](mailto:support@aras.com).

If Aras Dynamic Visualization (Dynamic 3D Viewer) was installed for the first time during the upgrade, Aras 3DV should be reconfigured after the upgrade as discussed in the *Installing Dynamic 3D Viewer When Upgrading* section.

#### Important

Please note that if the Streaming 3D Viewer is installed in an environment during upgrade where the Monolithic or Dynamic 3D Viewers have been previously installed and used, all existing native files of existing CAD Documents need to be re-converted. There is no automated means to perform this reconversion.



For more information on installing Streaming Viewer see *Automated Aras 3DV Installation for Streaming Viewer* section in the *Aras 3D Visualization 35 - Installation Guide*.



## Post-Upgrade Code Tree Cleanup Operations

Once the automated upgrade is completed, there may remain older files still in the Code Tree that should be deleted if found.



## Conversion Server File Cleanup

A few older HOOPS files may still be present in the Conversion Server folder.

These files can be found in the ConversionServer\bin folder inside the Innovator root folder:

1. Navigate to the folder: <Innovator\_root\_path>\ConversionServer\bin
2. If found, delete the following files:
  - o HoopsExchangeManager.dll
  - o HoopsExchangeManager.exp
  - o HoopsExchangeManager.lib



## Client Library File Cleanup

Older files from the HOOPS Web Viewer libraries may still be present, and can be deleted:

1. Navigate to the folder:

```
<Innovator_root_path>\Innovator\Client\Solutions\3DV\Scripts\3rdPartyLibs\hoopswebviewer
```

2. If found, delete the following files:

- o \hwf\hprt.hwf
- o \is\communicator\_server\_integration.js
- o \is\engine.wasm
- o \is\engine-asm.js
- o \is\engine-wasm.js
- o \is\hoops web viewer.js
- o \is\hoops web viewer\_monolithic.js
- o \is\hwf import.js
- o \is\iscroll.min.js
- o \is\jquery.minicolors.min.js
- o \is\jquery.ui.touch-punch.min.js
- o \is\jquery-3.5.1.min.js
- o \is\jquery-ui.min.js
- o \is\keymaster.min.js
- o \is\mobile-detect.min.js
- o \is\sample.js
- o \is\web viewer ui.js
- o \hoops web viewer sample.html
- o \hwf\_example.html



## Post-Upgrade Modification for the Streaming Viewer

Parameters for the Streaming Viewer have changed and need to be adjusted from prior versions, it is necessary to correct these parameters, located in the `ConversionServerConfig.xml` file.

### Important

If the streaming viewer was not installed/updated, then the following steps can be skipped.

1. Navigate to the root folder of the Innovator Code Tree (by default: `C:\Program Files(x86)\Aras\Innovator` ) and then locate and edit the file: `ConversionServerConfig.xml`
2. Modify the following argument values in the **Command arguments**, by ensuring the highlighted values are in place:
 

```
<ArasCadConverter>
  <Application converterPath="C:\H00PS Converter\bin\converter.exe" />
  <Command arguments="--sc compute bounding boxes 'All' --input ndf template file 'C:\Aras\
Innovator Server Name\H00PS Converter\Templates\Blank Template 1.ndf' --output ndf
'%filepath%\%filename%.ndf' --output png '%filepath%\%filename%.png' --output png resolution
'150x150' --output sc '%filepath%\%filename%' --sc create scz 'true' --sc compress_scz
'false' --output xml assemblytree '%filepath%\%filename%.xml' --output nrc
'%filepath%\%filename%.nrc' --background_color '1.0, 1.0, 1.0' --output_logfile
'%filepath%\%filename%.log'" />
```
3. Modify the following argument values in the **AssemblyCommand arguments** by adding the highlighted arguments:
 

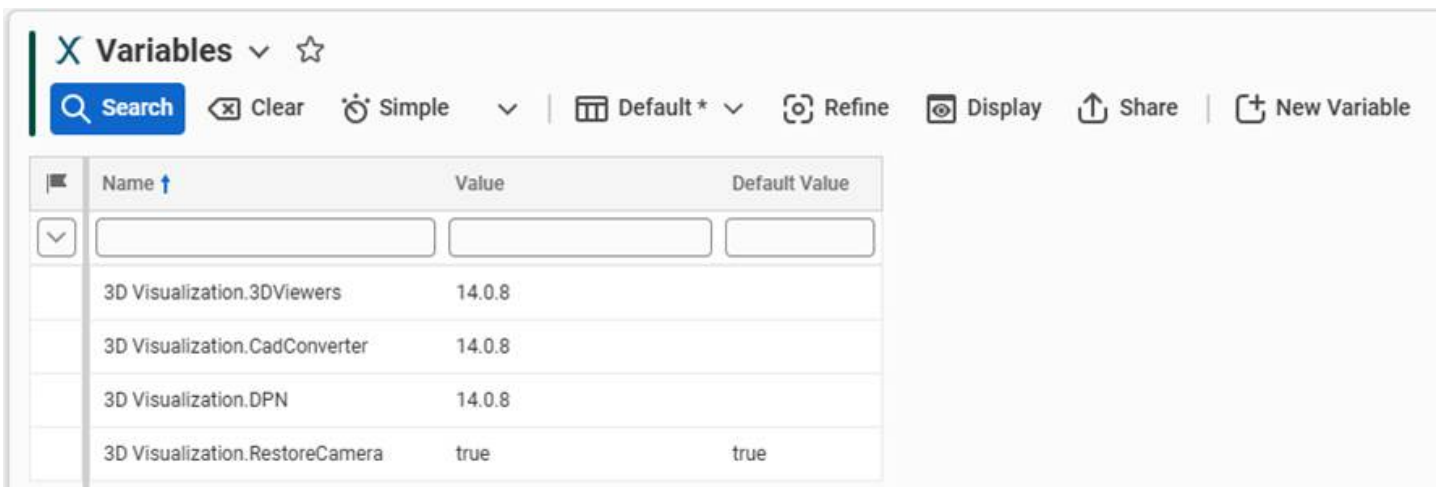
```
<AssemblyCommand arguments="--sc compute bounding boxes 'All' --input ndf template file
'C:\Aras\Innovator Server Name\H00PS Converter\Templates\Blank Template 1.ndf' --output ndf
'%filepath%\%filename%.ndf' --output png '%filepath%\%filename%.png' --output png resolution
'150x150' --sc create scz 'true' --sc compress_scz 'false' --output xml assemblytree
'%filepath%\%filename%.xml' --output nrc '%filepath%\%filename%.nrc' --background_color '1.0,
1.0, 1.0' --output_log file '%filepath%\%filename%.log'" streamingEnabled="True"/>
```
4. Save and close the file.



## Successful Upgrade Confirmation

The following steps outline the process to confirm the successful upgrade of given Aras 3D Visualization components:

1. Log into Aras Innovator as an administrator.
2. From Table of Contents, select **Administration**.
3. Select **Variables**.
4. Confirm that Variables for the given components are listed with necessary Values:
  - o 3D Visualization.3DViewers: 14.0.8
  - o 3D Visualization.CadConverter: 14.0.8
  - o 3D Visualization.DPN: 14.0.8
  - o 3D Visualization.RestoreCamera: true



The screenshot shows the 'Variables' page in Aras Innovator. The page has a search bar and several filters: 'Simple', 'Default \*', 'Refine', 'Display', 'Share', and 'New Variable'. Below the filters is a table with the following data:

Name ↑	Value	Default Value
3D Visualization.3DViewers	14.0.8	
3D Visualization.CadConverter	14.0.8	
3D Visualization.DPN	14.0.8	
3D Visualization.RestoreCamera	true	true

If there is no Variable with the correct Value for a component, the upgrade of this component fails. Restore the Aras Innovator code tree, Conversion Server, and database with the backups done before the upgrade and contact Aras Support at [support@aras.com](mailto:support@aras.com) .

# User Guide



## Introduction

### [Purpose](#)

This User Guide describes how to use the Aras 3D Visualization (3DV) platform component within Aras Innovator.



## Scope

This document provides instructions to define, manage, and validate all 3D Visualization (3DV) features. This Guide does not cover any custom or customized Aras 3DV functionality.



## Target Audience

This User Guide is intended for Aras 3D Visualization (3DV) users responsible for utilizing the features outline in this document.



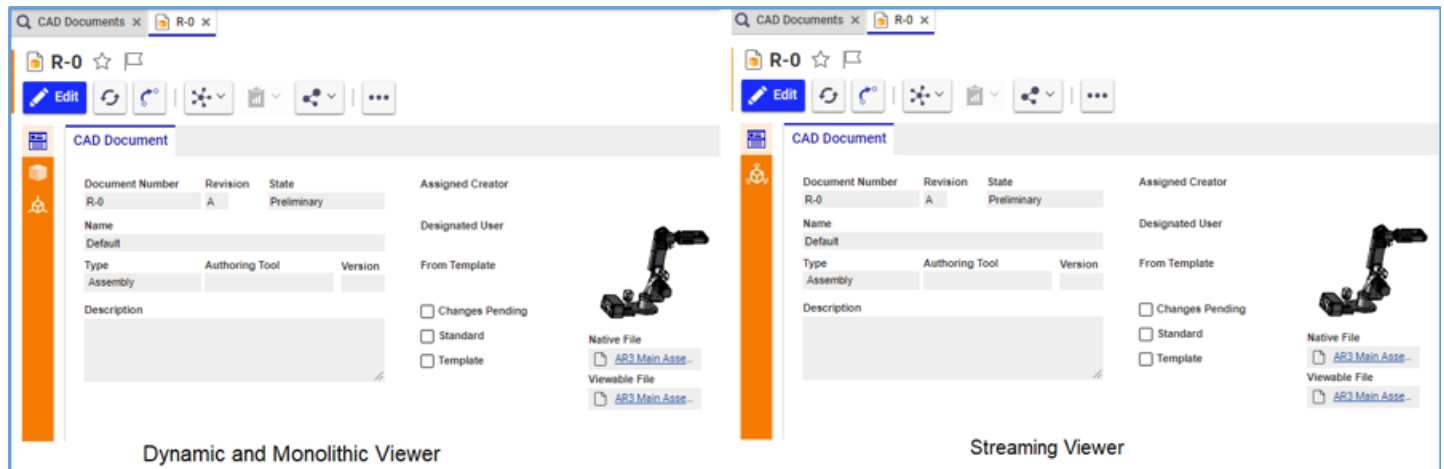
## Introduction to Aras 3D Visualization (3DV)

Aras 3D Visualization (3DV) is an optional component for the Aras Innovator platform. It is a business-ready solution for converting and visualizing native 3D computer-aided design (CAD) files in Aras Innovator. Users upload 3D models from a CAD authoring tool into Aras Innovator as native 3D CAD files. The system automatically converts the uploaded native files into Stream Cache Single (SCS), Stream Cache Compressed (SCZ), 3D PDF viewable files, and other optional output formats. The users can view, manipulate, manage, and visually collaborate on the 3D models from the viewable files.



## CAD Documents and 3DV

In Aras Innovator, a **CAD Document** Item represents a part or assembly drawing, specification, 3D model, and so on. For Aras 3DV visualization purposes, the **CAD Document** Item should be a mechanical part or assembly 3D model created in a CAD editor.



Hereinafter, a CAD Document will mean a **CAD Document** Item to simplify wording.

An assembly 3D model shows a complex product that consists of multiple parts (components). In such a case, a given CAD Document includes one or more other CAD Documents on its **Structure Relationships** tab. Another industry-common term is a parent CAD Document that has child CAD Documents. In technical terms, an assembly **CAD Document** Item has a **CAD Document Structure Relationship** Item for each included part **CAD Document** Item, where the assembly **CAD Document** Item is a source Item, and a part **CAD Document** Item is a related Item. A child CAD Document can be a leaf part or another parent (subassembly). The latter case is a multi-level CAD (Document) structure or assembly.

R-0 ☆ □

Edit ↻ ↺ ⚙️ 🗑️ 🔗 ⋮

☰ CAD Document

☰ Structure Parents Files Changes

📁 CAD Documents ☆

⚙️ 🔍 🗑️ Hidden ▾ 📷 🖨️ 🔗

Seque...	Document Number ↑	R.	Name	Type	S.	Native File...	Viewable File ...	Authoring Tool	Dependency	C...	S...	T...	Dynamic Enabled
128	R-1	A	Base Assembly-2	Mechanical/Assembly	P...	<a href="#">Base...</a>	<a href="#">Base Ass...</a>			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
384	R-10	A	Limit Switch SV-166-1C25-1	Mechanical/Part	P...	<a href="#">Limit...</a>	<a href="#">Limit Swit...</a>			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
512	R-11	A	Bearing 32009X-1	Mechanical/Part	P...	<a href="#">Bearin...</a>	<a href="#">Bearing_3...</a>			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

For more information on CAD Documents, see the *Aras Innovator Product Engineering 31 - User Guide*.




## Managing 3D CAD Files

A CAD Document can have files associated with it. To show a 3D model of a part or assembly in Aras 3DV, the **Native File** property of a CAD Document should include a native CAD file with this 3D model. One CAD Document can have only one native CAD file.



### Native File

 [AR3 Main Asse...](#)

### Viewable File

 [AR3 Main Asse...](#)


A CAD Document can also contain a 3D PDF viewable file with the 3D model in the **Viewable File** property. The CAD Document Item form shows a link to this 3D PDF file.



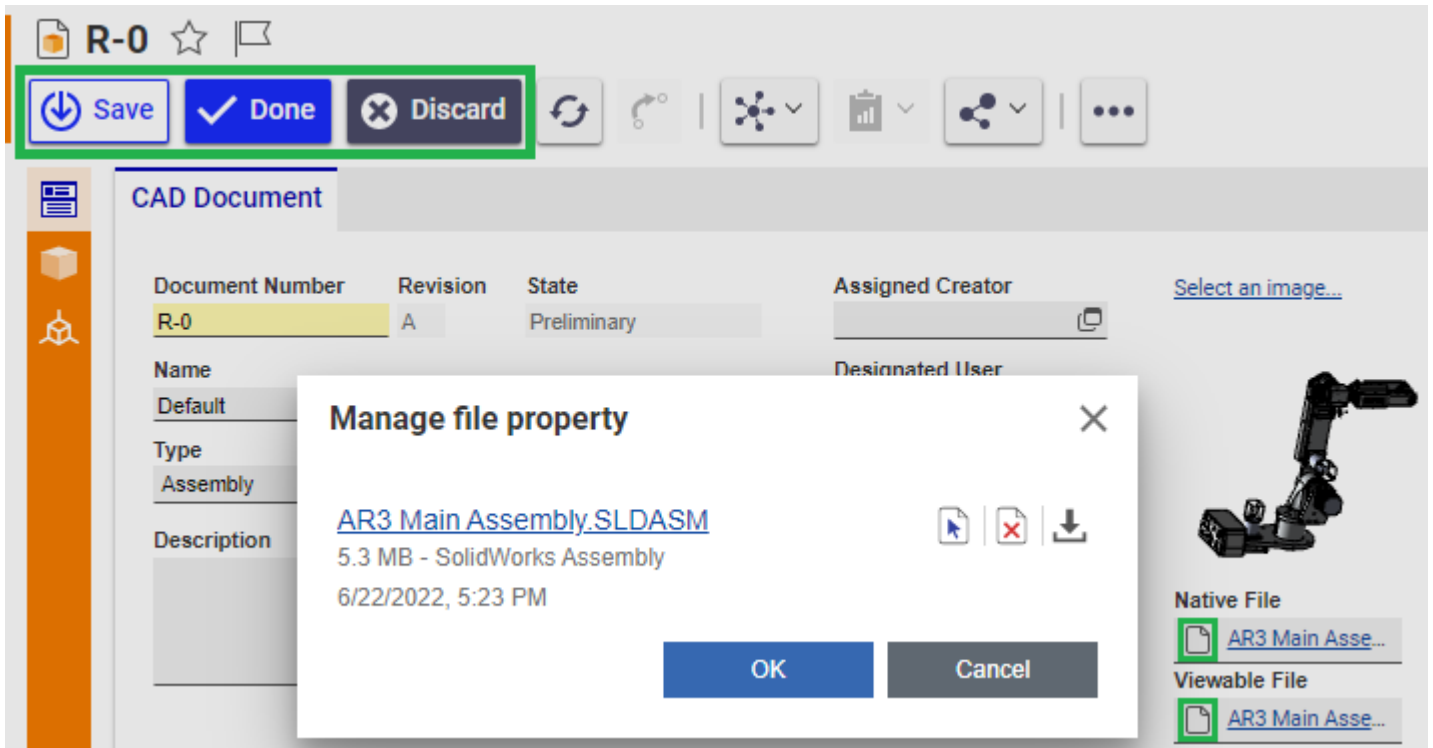
### Native File

 [AR3 Main Asse...](#)

### Viewable File

 [AR3 Main Asse...](#)

End users with the Permission to edit a CAD Document in a given Item State can attach, download, replace, and remove native CAD and viewable 3D PDF files with 3D models to the **Native File** and **Viewable File** CAD Document properties using the standard Aras Innovator functionality provided with the **Manage file property** dialog. For more information on file management, see the *Aras Innovator Product Engineering 14 - User Guide*.



Additionally, admins and advanced users can import multiple native CAD files with 3D models into Aras Innovator with the Aras Batch Loader tool. CAD Documents and CAD Document Structures can be automatically created during the import. This approach is efficient for uploading large CAD assemblies into Aras Innovator. For details about this tool and its requirements, see *Aras Innovator 21 - Batch Loader*.

Third-party connectors for CAD authoring tools can be an efficient way for end-users to manage adding and updating CAD Documents.

After a native CAD file is attached to a CAD Document, the Aras CAD Converter automatically schedules this file for conversion into the following three viewable files:

- SCS (Stream Cache Single)
- SCZ (Stream Cache Compressed)
- 3D PDF (Portable Document Format)

Once the conversion is successfully completed, the Aras CAD Converter attaches these files to the CAD Document. The Aras CAD Converter is a separate Aras 3DV component.

End users with the Permission to get and access CAD Documents can view 3D models from:

- SCS files using the Monolithic or Dynamic Viewer, which are separate Aras 3DV components.
- SCZ files using the Streaming Viewer.
- 3D PDF files using third-party PDF viewers and editors not included in Aras 3DV.

When a native CAD file with a 3D model is removed from a CAD Document, end users will not be able to view this 3D model because Aras 3DV automatically removes relationships between this CAD Document and a pertinent viewable SCS file.

Viewable SCS files are fully automatically managed by Aras 3DV: end users cannot access or manage them directly.



## Accessing 3DV Functionality

Aras 3DV functionality is accessible from a CAD Document Item form as follows:

The screenshot shows the 'CAD Document' form with the following fields and options:

- Document Number:** R-0
- Revision:** A
- State:** Preliminary
- Assigned Creator:** (empty)
- Designated User:** (empty)
- Name:** Default
- Type:** Assembly
- Authoring Tool:** (empty)
- Version:** (empty)
- From Template:** (empty)
- Description:** (empty text area)
- Changes Pending
- Standard
- Template
- Native File:** AR3 Main Asse...
- Viewable File:** AR3 Main Asse... (highlighted with a red box and number 4)

On the left sidebar, three icons are highlighted with red boxes and numbers 1, 2, and 3, corresponding to the Monolithic, Dynamic, and Streaming viewers respectively.

- The **Monolithic Viewer** sidebar. This viewer displays the Monolithic view of the selected CAD item.
- The **Dynamic Viewer** sidebar. This viewer displays the Dynamic view of the selected CAD item.
- The **Streaming Viewer** sidebar. This viewer also displays the Dynamic view of the selected CAD Item using an alternate view generation framework.
- The **Viewable File**. This link enables user to download the 3D model as a 3D PDF file.

### Important

The Dynamic and Streaming Viewers have the similar functionality; however, the viewers available will depend on the installation of Aras 3DV.

### Important

Please note that the output files of Dynamic/Monolithic Viewer and Streaming Viewer are incompatible with one another.

If the Streaming 3D Viewer is installed in an environment where the Monolithic or Dynamic 3D Viewers have been previously

installed and used, all existing native files of existing CAD Documents need to be re-converted. There is no automated means to perform this reconversion.



## Monolithic vs. Dynamic Viewer

Both Monolithic and Dynamic Viewers show a 3D model in Aras Innovator by rendering 3D component geometries stored in viewable SCS files. The difference between them is in the following:

- Which 3D models they can show.
- How they populate the view.
- What limitations they have.
- What functions they offer.

The Monolithic Viewer is the core 3D Visualization functionality and a prerequisite for the Dynamic Viewer. It can show a static 3D model of a component or assembly. Technically, this Viewer shows one viewable SCS file attached to a given component. In an assembly case, such a file is created from the files attached to all sub-assemblies and parts included in this assembly when it was loaded into the Viewer. The shown 3D model is a static assembly view because it cannot contain geometry changes to components made after the assembly view file had been created. Thus, the Monolithic Viewer features basic functionality for viewing the 3D model, isolating, and hiding assembly components.

The Dynamic Viewer provides optional 3D Visualization functionality in addition to the Monolithic Viewer. It can show a dynamic 3D model of an assembly only. Technically, this Viewer shows multiple viewable SCS files attached to all sub-assemblies and parts included in this assembly when it was loaded into the Viewer. Since a given model is a set of SCS files, displaying product manufacturing information is not included. The shown 3D model is a dynamic assembly view because it can contain geometry changes to components made after the assembly view file was created according to various structure resolutions: Latest, Released, and so on. Thus, in addition to the functionality included in Monolithic Viewer, the Dynamic Viewer has capabilities for manual geometry transformations and annotations. Aras Innovator Admins can also customize what content is displayed and how it is displayed in the Dynamic Viewer. Such customization is beyond the scope of this User Guide.

This Guide describes both Viewers in detail in the corresponding sections.



## Dynamic vs. Streaming Viewer

The Dynamic and Streaming Viewers have similar functionalities. The differences are as follows:

- The Dynamic and Streaming Viewers use different, non-compatible view files (SCS, and SCZ respectively).
- For Streaming Viewer, a separate window service called Hoops Server is used. The HOOPS Server initiates individual Stream Services to support each Streaming Viewer instance.
- Larger datasets can be streamed from a server to a client in a single request with the help of the Streaming Viewer unlike Dynamic Viewer which uses individual client/server HTTP requests.

### Important

Please note that the output files of Dynamic/Monolithic Viewer and Streaming Viewer are incompatible with one another.

If the Streaming 3D Viewer is installed in an environment where the Monolithic or Dynamic 3D Viewers have been previously installed and used, all existing native files of existing CAD Documents need to be re-converted. There is no automated means to perform this reversion.



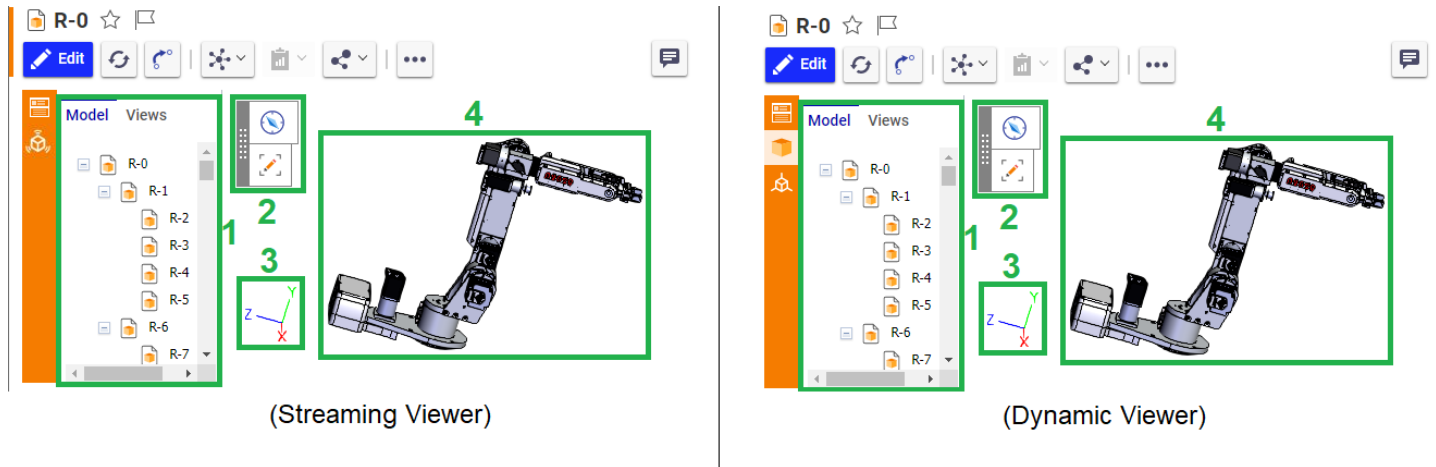
## Aras 3DV UI

This section describes Aras 3DV UI elements general for the Monolithic, Dynamic and Streaming Viewers: the 3D scene, 3DV toolbars, and context menus.



## 3D Scene

A 3D scene is the same for Monolithic, Dynamic and Streaming Viewers.



The 3D Scene includes the following:

1. The **Model Browser** section. As its contents and visibility are different for the Viewers, it is discussed in a corresponding section for each Viewer.
2. The **3DV** toolbar.
3. The **Model Orientation** indicator.
4. A given 3D CAD model.



## Aras 3DV Toolbar

The Monolithic, Dynamic and Streaming Viewers use the same **3DV** toolbar, which is dynamic and flexible. When opening any of the Viewers on a newly opened CAD Document, this toolbar is collapsed and located in the upper left corner of a 3D scene view.



The collapsed **3DV** toolbar has the following elements:



1. The **Movement** bar: dragging this bar moves the **3DV** toolbar through the 3D scene.
2. The **View** button: clicking this button switches the 3D scene to the 3D model viewing mode and either shows or hides the **Basic Viewing** toolbar; see the Viewing Toolbars section.
3. The **Markup** button: clicking this button switches the 3D scene to the 3D model markup mode and either shows or hides the **Basic Markup** toolbar; see the Markup Toolbars section.

A larger button indicates that the 3D scene is currently in a mode launched by this button. For example, if the **View** button is larger than the **Markup** button, the scene is in the viewing mode.

## Viewing Toolbars

Clicking the **View** button on the collapsed **3DV** toolbar launches the viewing mode and either shows or hides the **Basic Viewing** toolbar, which includes the following buttons:



1. **Zoom Up** increases a 3D CAD model size on the canvas along the viewpoint center.
2. **Zoom Down** decreases a 3D CAD model size on the canvas along the viewpoint center.
3. **Reset View** sets the default 3D CAD model orientation and size on the canvas.
4. **Switch To Standard Toolbar** hides the **Basic Viewing** toolbar and shows the **StandardViewing** toolbar.

The **Standard Viewing** toolbar is fixed to the upper left corner of the 3D scene view: it cannot be moved through the 3D scene like the **Basic Viewing** toolbar. The **Standard Viewing** toolbar includes the same buttons as on the **Basic Viewing** toolbar with the exception of the **Switch To Standard Toolbar** button (4)—the **Switch To Basic Toolbar** button appears instead (15)—and the following additional items:



5. **Preferences** shows the **3D Viewer Preferences** pop-up dialog box for configuring the projection and zoom preferences; see the *3D Viewer Preferences* section.
6. **Zoom Window** enables a user to select an arbitrary area on the model to zoom into this area; see the *Zooming into Area* section.
7. **Orient to Face** shows a given 3D CAD model zoomed into a selected surface or part and oriented to a user; see the *Zooming into Part* section.
8. **Set Display Style** shows or hides a toolbar with options on how to display a given 3D CAD model; see the *3D Viewer Preferences* section.
9. **Measure** shows or hides a toolbar with options for measuring a given 3D CAD model; see the *Measuring CAD Models* section.
10. **Exploded View** shows or hides a toolbar with the slider exploding a given 3D CAD model; see the *Exploding CAD Models* section.
11. **Cross Section** shows or hides a toolbar with options for cross-section viewing of a given 3D CAD model; see the *Cross Section Viewing of the CAD Model* section.



12. **Model Browser** shows or hides the **Model Browser** section with quick viewing options as trees. The Monolithic have different **Model Browser** sections than in the Dynamic or Streaming Viewer; see the *Model Browser in Monolithic Viewer* and *Model Browser in Dynamic and Streaming Viewer* sections.
13. **Product Manufacturer Information** displays annotations and attributes contained in a 3D solid model that complement the geometric definition and complete the digital data set; see the *Product Manufacturing Information* section. This button is visible only in the Monolithic Viewer.
14. **Configurations** is the list of configurations available within the given 3D CAD model.
15. **Export SVG** enables a user to export an SVG Image file of the current 3D View to the local machine. See the *Exporting SVG Image* file section. This is only available for Dynamic and Streaming Viewer.
16. Switch to Basic Toolbar hides the Standard Viewing toolbar and shows the Basic Viewing toolbar.



## Markup Toolbars

Clicking the **Markup** button on the collapsed **3DV** toolbar launches the markup mode and either shows or hides the **Basic Markup** toolbar, which includes the following buttons:



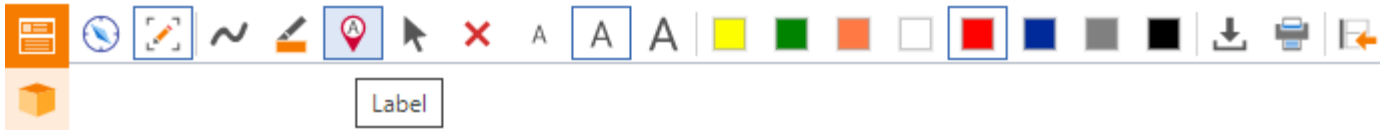
1. **Scribble** shows or hides buttons on the toolbar for drawing arbitrary lines on a given 3D CAD model; see the *Drawing on CAD Models* section.
2. **Highlight** shows or hides buttons on the toolbar for drawing rectangular areas on a given 3D CAD model; see the *Highlighting Areas on CAD Models* section.
3. **Label** shows or hides buttons on the toolbar for adding labels to a given 3D CAD model; see the *Putting Labels on CAD Models* section.
4. **Delete** removes selected markup items from a given 3D CAD model; see the *Deleting Markups on CAD Models* section.
5. **Switch To Standard Toolbar** hides the **Basic Markup** toolbar and shows the **Standard Markup** toolbar.

The **Standard Markup** toolbar is fixed to the upper left corner of the 3D scene view: it cannot be moved through the 3D scene like the **Basic Markup** toolbar. The **Standard Markup** toolbar includes the same buttons as on the **Basic Markup** toolbar with the exception of the **Switch To Standard Toolbar** button (5)—the **Switch To Basic Toolbar** button appears instead (9)—and the following additional buttons:



6. **Select** selects **markups** on a given 3D CAD model.
7. **Download File** downloads a JPG image of a given markup.
8. **Print** uses the standard browser print function for markups. Depending on a given browser and its specific configuration, various print choices, such as printing on a printer or saving to PDF, are available.
9. **Switch to Basic Toolbar** hides the Standard Markup toolbar and shows the Basic Markup toolbar.

Unlike the **Basic Markup** toolbar, the **Standard Markup** toolbar is dynamic. The **Standard Markup** toolbar does not show or hide an additional toolbar containing functionality for a markup feature selected with the **Scribble** (1), **Highlight** (2), or **Label** (3) button. It displays the functionality of a selected markup feature. A selected feature and functionality options are enclosed by blue boxes. To hide the functionality options, click the selected feature.



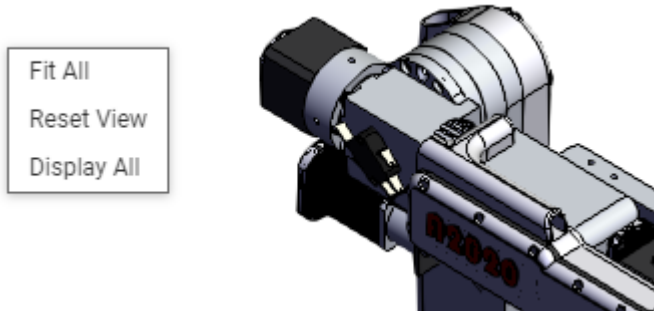
## Aras 3DV Context Menus

The Monolithic, Dynamic and Streaming Viewers feature context menus for the 3D scene and a selected part. Both **3D Scene** and **Part** context menus are available only in the model viewing mode. Menu content depends on a given Viewer. This section discusses the menu options general for both Viewers.

### 3D Scene Context Menu

To access the **3D Scene** context menu, right-click either:

- The canvas outside a given 3D CAD model.



- A given 3D CAD model with no selected parts.



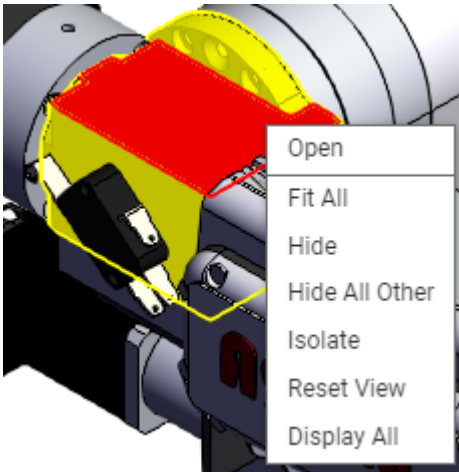
The **3D Scene** context menu includes the following commands:

- Fit All sets the default 3D CAD model size so that all visible model parts are shown on the 3D scene.
  - Reset View makes all 3D CAD model parts visible on the 3D scene (shows them) and sets the given 3D CAD model to its default orientation and size on the 3D scene.
  - Display **All** makes all 3D CAD model parts visible on the 3D scene (shows them).
- The Dynamic and the Streaming Viewer **3D Scene** context menu has additional commands; see the 3D Scene Context Menu in Dynamic Viewer and Streaming Viewer section.

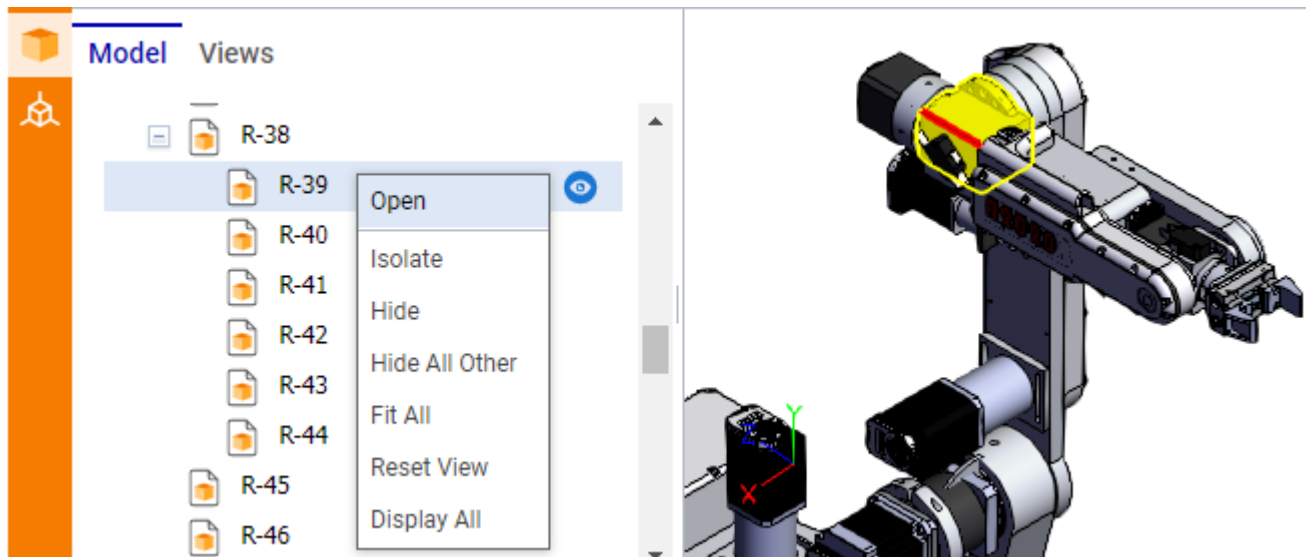
## Part Context Menu

To access the **Part** context menu, click an individual part to select it and then right-click either:

- On a given 3D CAD model on the canvas.



- In the Model Browser.



In addition to the **Fit All**, **Reset View**, and **Display All** commands from the **3D Scene** context menu, the **Part** context menu includes the following commands:

- Open opens the CAD Document Item view of a selected part.

- Isolate makes everything transparent except the selected part and zooms into this part; see the *Zooming into and Isolating Part* section.
- Hide makes a selected part invisible on the 3D scene (hides it).
- Hide **All Other** makes unselected parts invisible on the 3D scene (hides them).

The Dynamic and Streaming Viewer **Part** context menu has additional commands. It is possible to select multiple individual parts in the Dynamic/Streaming Viewer. See the *Part Context Menu in Dynamic Viewer and Streaming Viewer* section.

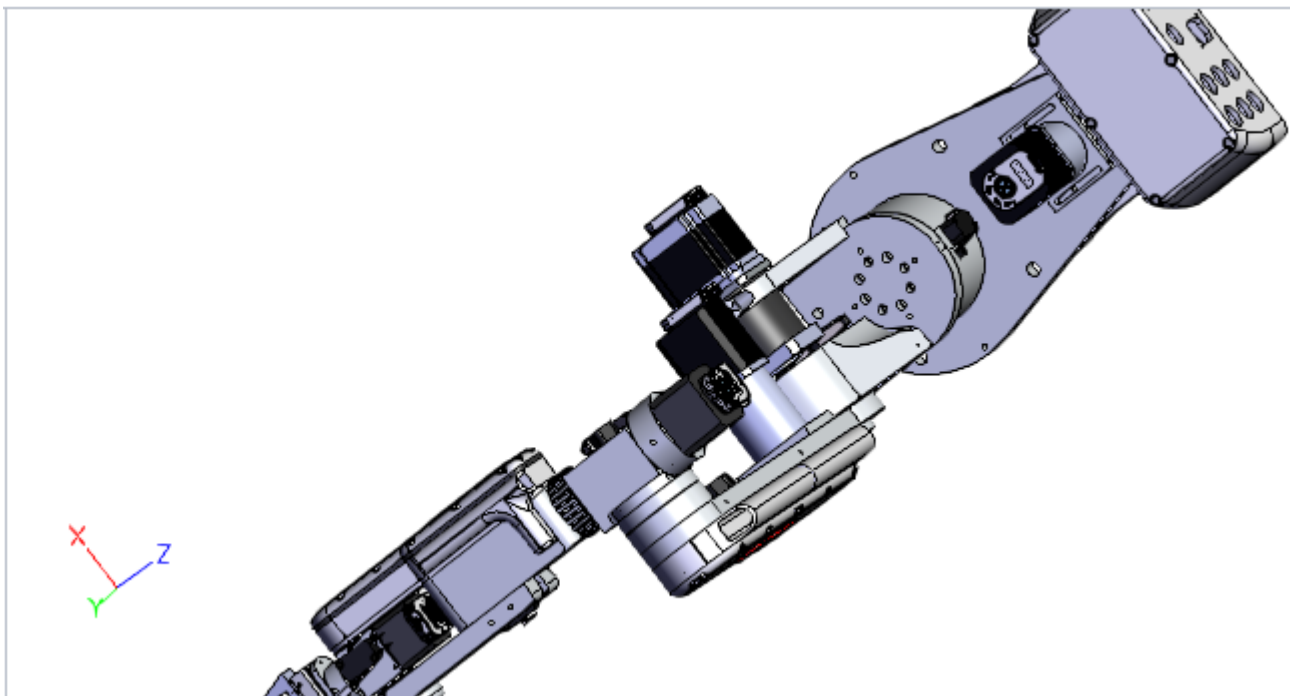
## General 3DV functionality

This section describes the Aras 3DV functionality general for the Monolithic, Dynamic and Streaming Viewers. This functionality enables end users to view and annotate 3D CAD models.

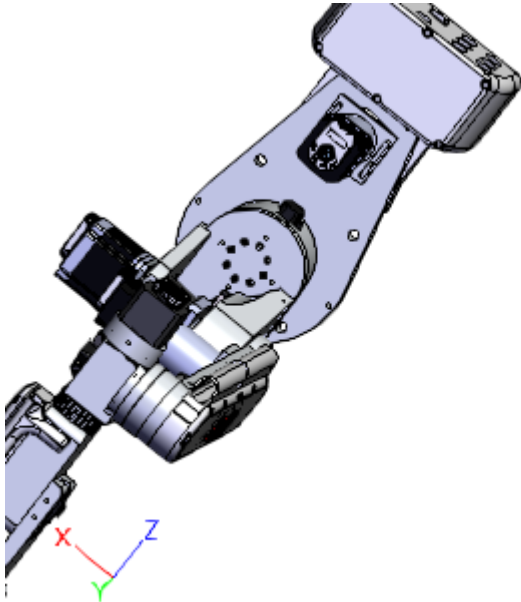


## Viewing CAD Models

In the 3D model viewing mode launched with the **View** button on the **Basic Viewing** or **Standard Viewing** toolbar, an end user can view a 3D CAD model from different sides by clicking and dragging any place on the model or canvas. The **Model Orientation** indicator tells which side is toward the user and where the other sides are.

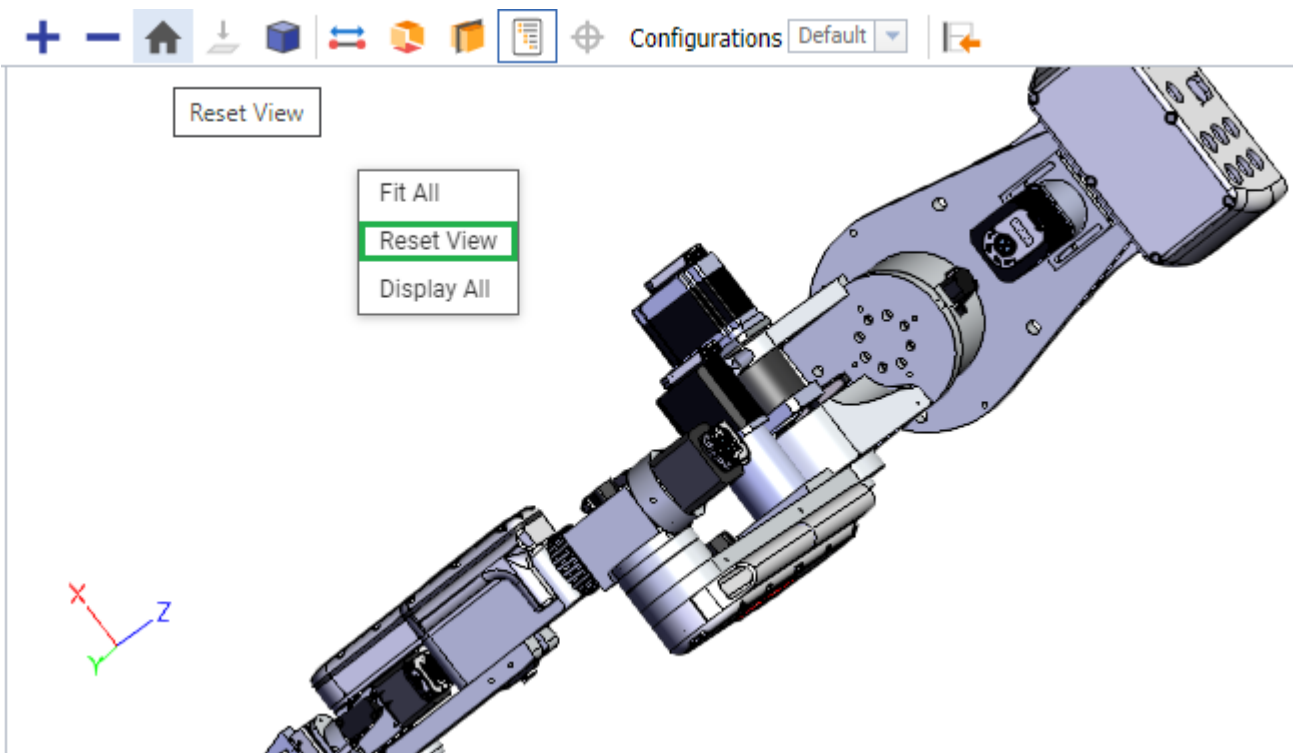


Right-clicking and dragging any place on the model or canvas pans the view—moves the model through the canvas without changing the model orientation.

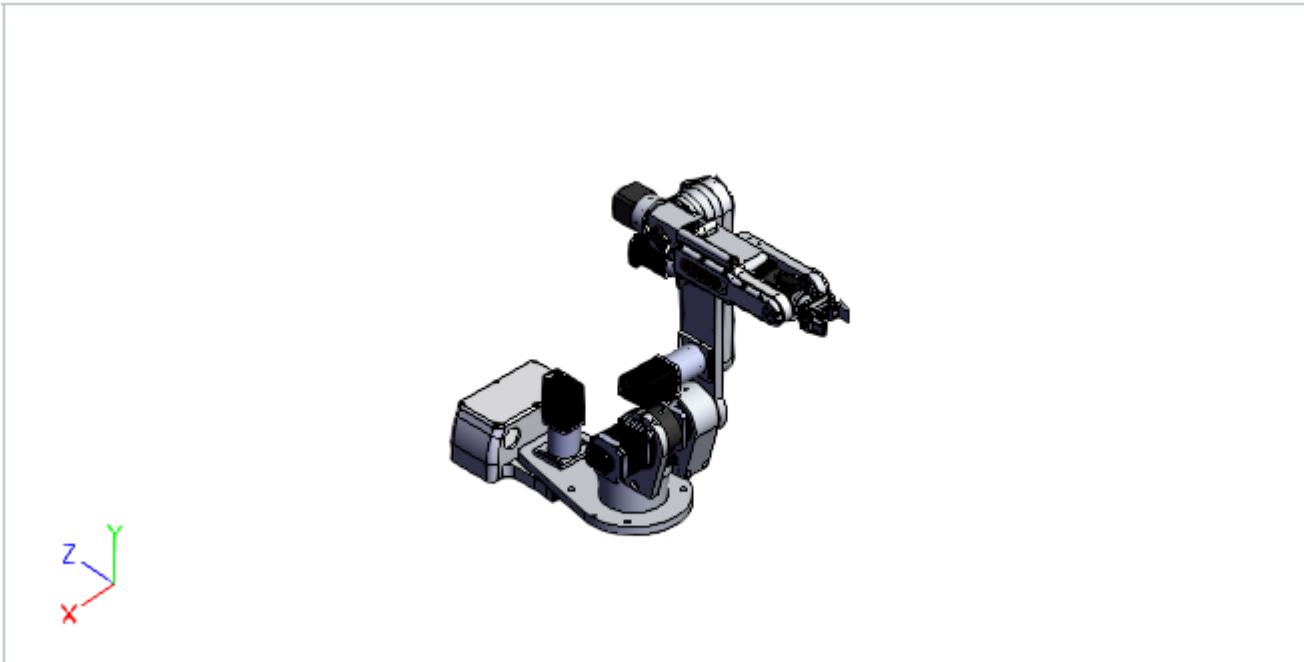


To show a 3D CAD model in its default orientation and size:

- Click Reset View on the Basic Viewing or Standard Viewing toolbar or 3D Scene or Part context menu.



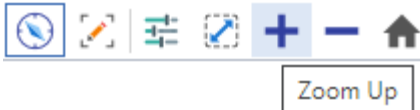
And the model will return to its default orientation and position in front of the camera.



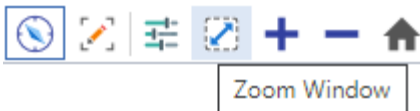
## Zooming

The zooming options are the following:

- Wheel mouse button: rotate this wheel button to zoom in or out according to the zoom configuration set in the **3D Viewer Preferences** dialog box; see the Zoom Preferences section.
- The **Zoom Up** and **Zoom Down** buttons on the **Basic Viewing** and **StandardViewing** toolbars: click the appropriate button to zoom in or out along the 3D scene viewpoint center.



- The **Zoom Window** feature available from the **StandardViewing** toolbar: click this button and select an arbitrary area on the 3D scene to zoom into this area; see the Zooming into Area subsection.



- The **Orient to Face** feature available from the **StandardViewing** toolbar: selects a surface and click this button to orient and zoom a given 3D CAD model into this part or surface; see the Zooming into Part subsection.



- The **Fit All** feature available from the **3D Scene** and **Part** context menus: click this command to restore the default 3D CAD model size fitting the 3D scene canvas without changing the current model orientation; see the Zooming into and Isolating Part subsection.



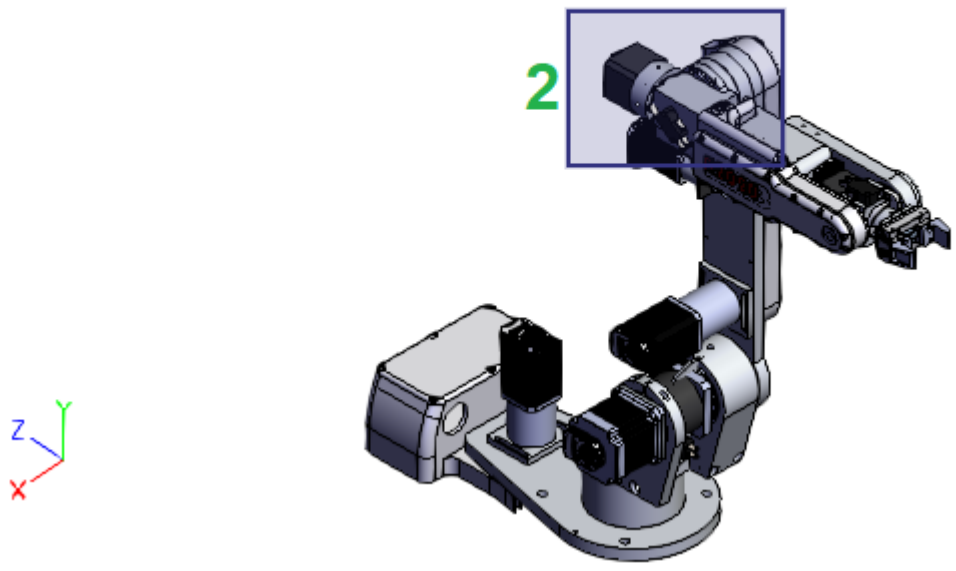
- The **Isolate** feature available from the **Part** context menu: select a part or surface and click this command to zoom into this part or surface and make all other parts and surfaces transparent; see the Zooming into and Isolating Part subsection.

- Open
- Fit All
- Hide
- Hide All Other
- Isolate
- Reset View
- Display All

### Zooming into Area

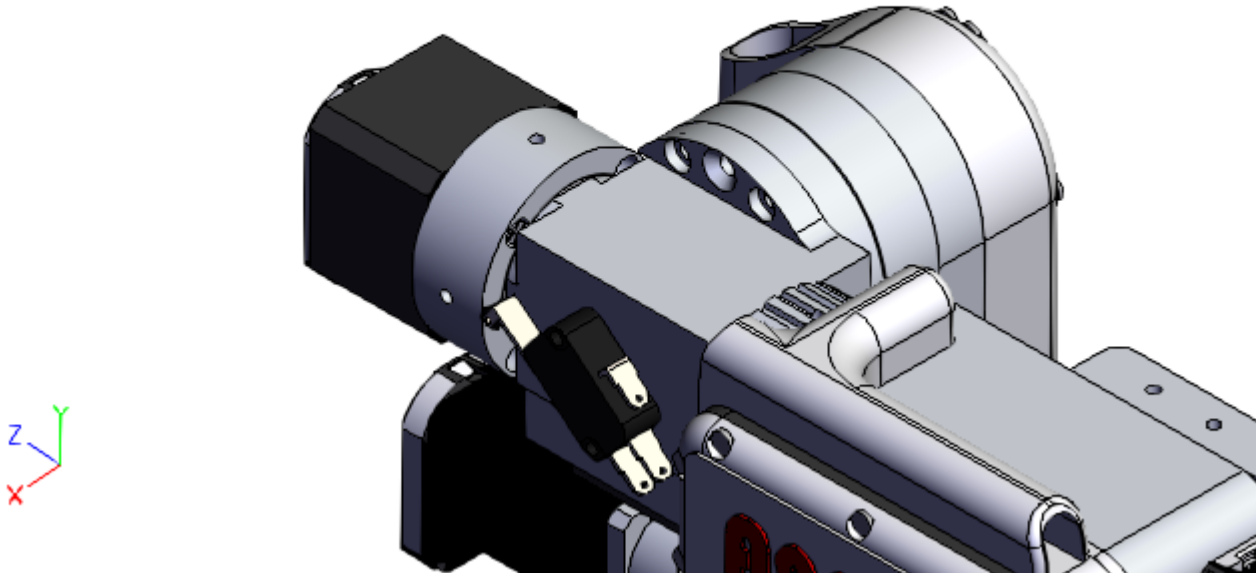
To zoom into an area on the 3D scene:

1. Click the Zoom **Window** button on the **StandardViewing** toolbar.
2. Select the area by clicking and dragging.



The selected area will fit the 3D scene canvas.





### Orienting Model to Face

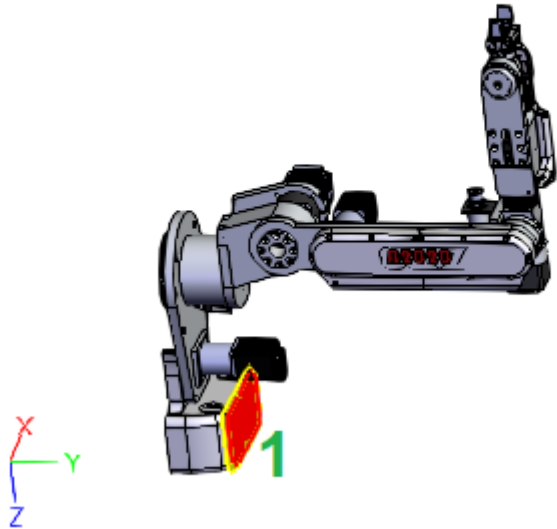
To orient a 3D CAD model to a surface on its part:

1. Select the part or surface
2. Click the Orient to Face button on the **StandardViewing** toolbar.

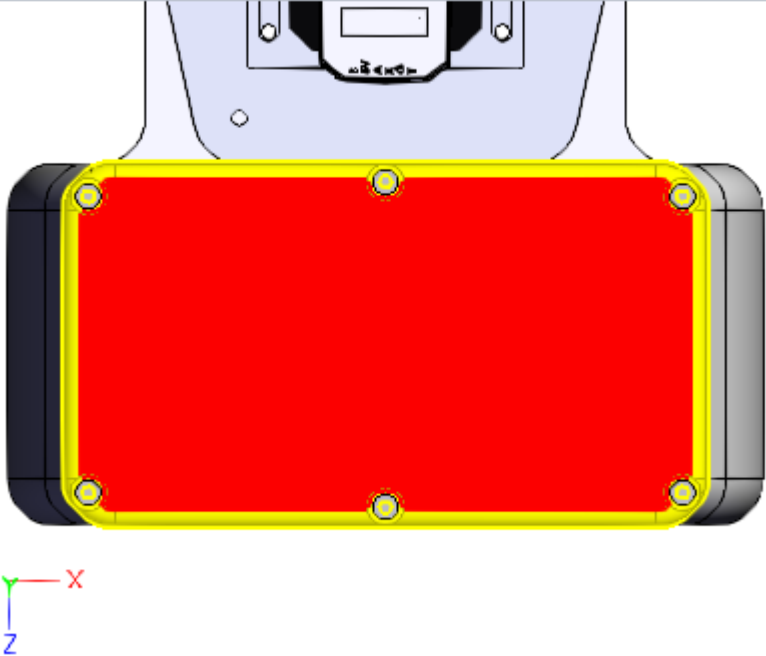




2



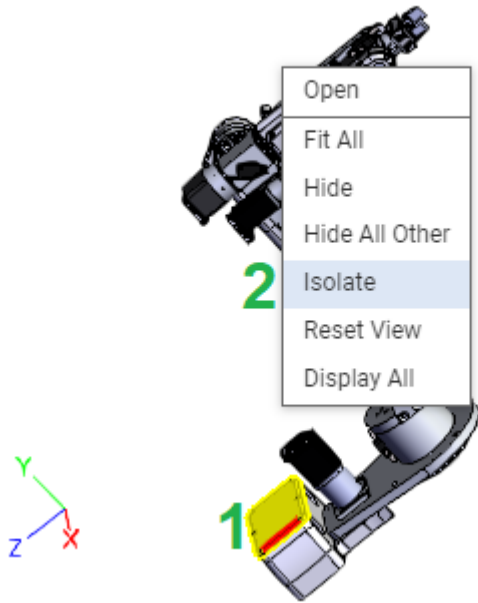
The selected part or surface will fit the 3D scene canvas and be oriented to the user.



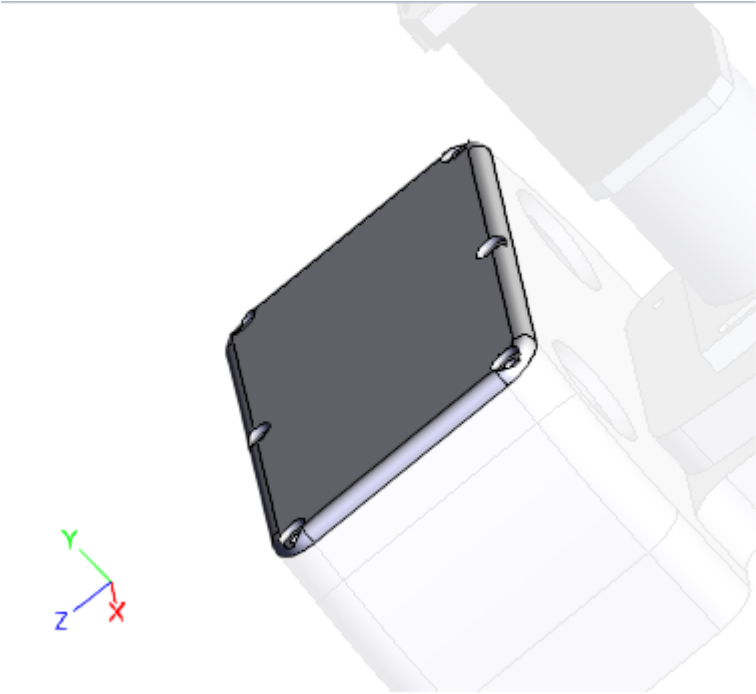
## Isolating Part

To zoom into a selected part in a 3D CAD model and make all other parts transparent:

1. Select the part or surface.
2. Right-click and click the Isolate command.



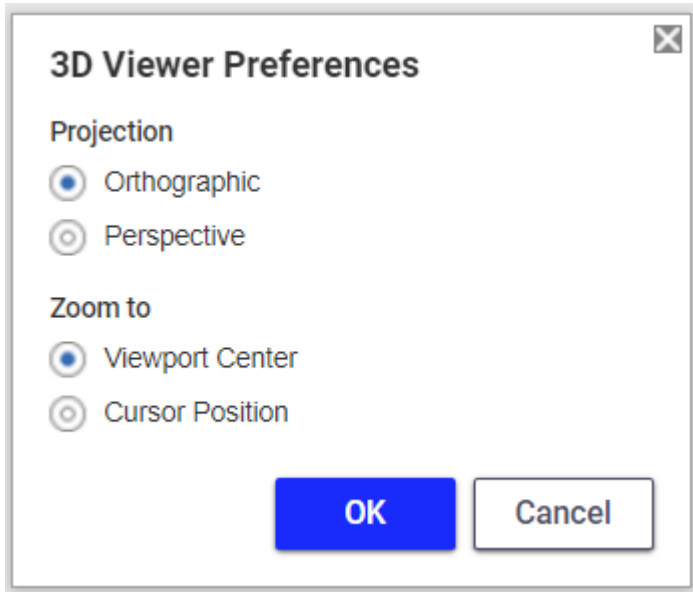
The selected part will fit the 3D scene canvas without changing the orientation, and all other parts will be transparent.



### 3D Viewer Preferences

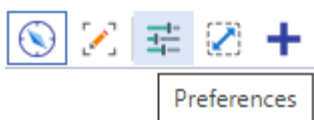
With the **3D Viewer Preferences** dialog, an end user can configure the following preferences for viewing:

- **Projection:** a 3D model projection mode.
- **Zoom to:** an alignment points for zooming with a wheel mouse button.



These configurations are persistent for a given Viewer and logged-in user. An end user can have different configurations for the Monolithic and Dynamic/Streaming Viewers.

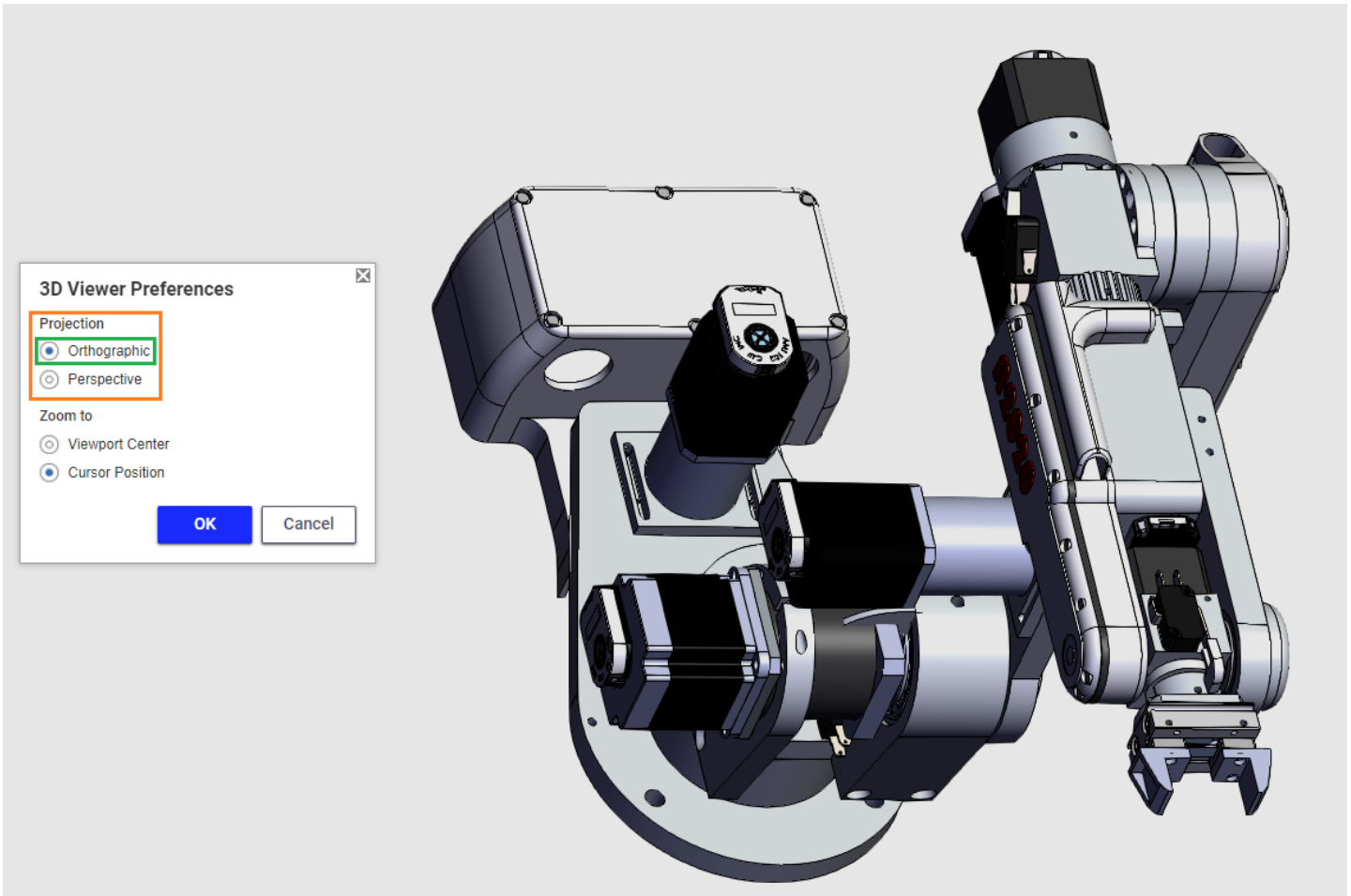
The **Preferences** button on the **Standard Viewing** toolbar provides access to the **3D Viewer Preferences** dialog.



### Projection Preferences

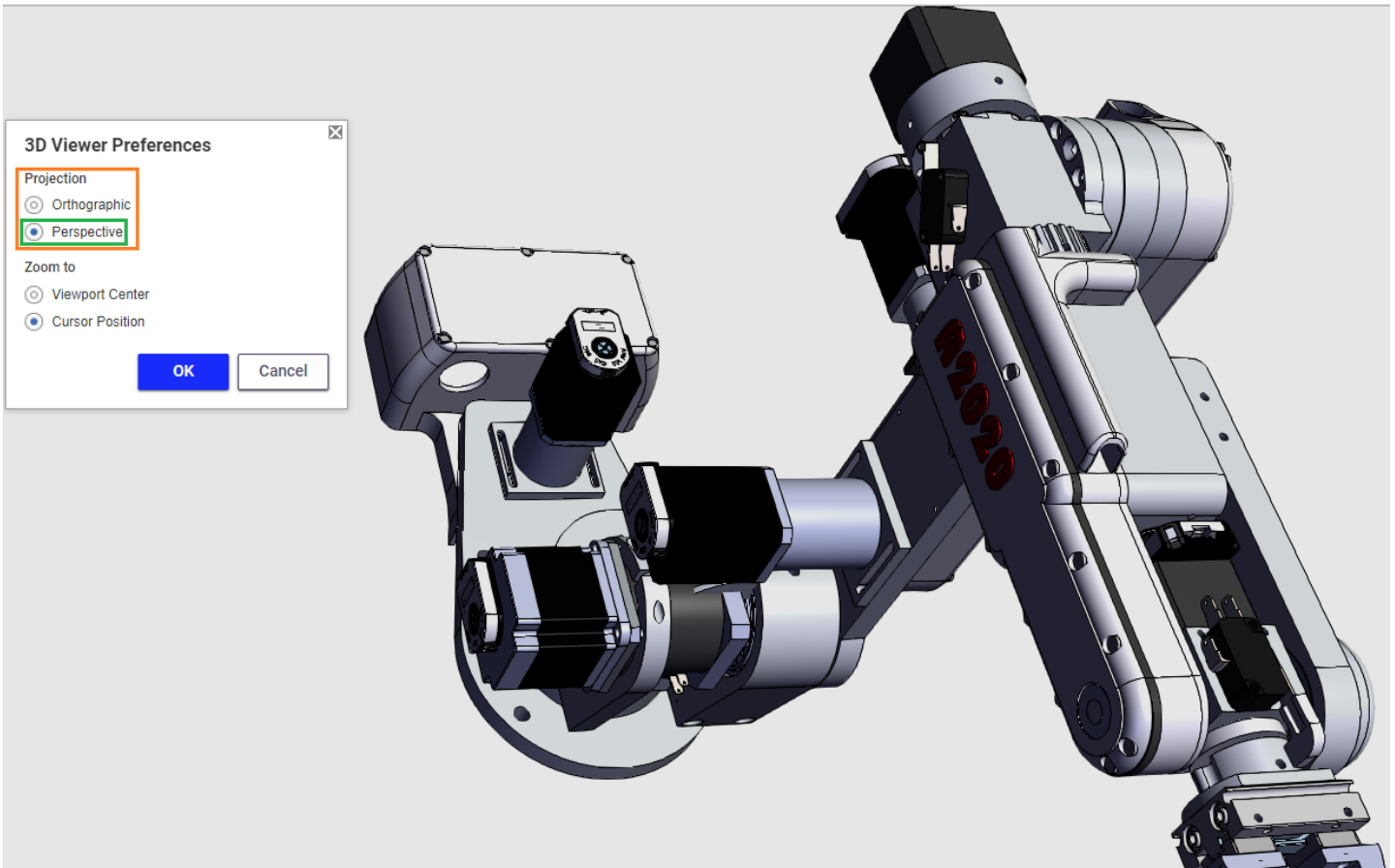
The **Projection** preference in the **3D Viewer Preferences** dialog offers the following options for projecting a 3D model on a scene:

**Orthographic:** to show all objects on a 3D scene at the same scale regardless of their distance to a user. The farthest and nearest objects have the same size. This setting is default and reflects the legacy orthographic view projection behavior.



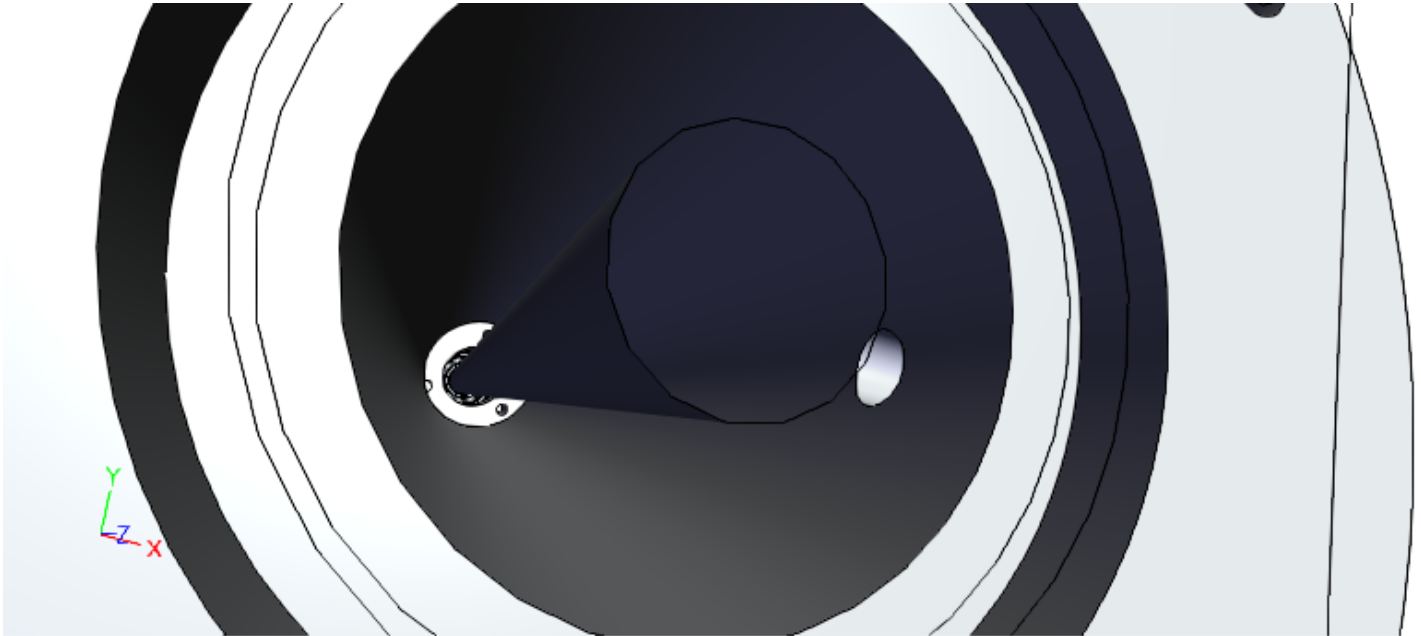
Perspective to show the following:

- To show all objects on a 3D scene at different scales correspondingly to their distance to a user. The farther objects have smaller sizes than the nearer ones.



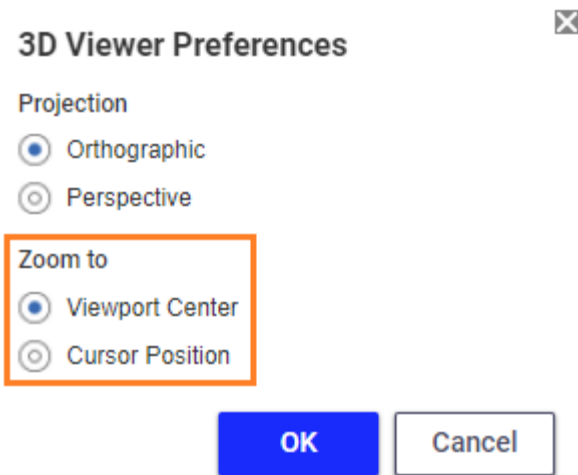
- To zoom through a model. A user can view the model from the inside.





### Zoom Preferences

The **Zoom** preference in the **3D Viewer Preferences** dialog offers the following options for zooming with a wheel mouse button:



- **Viewport Center:** to zoom in and out along the 3D scene viewpoint center regardless of the current cursor position. This setting is the default and reflects the legacy zooming behavior.
- **Cursor Position:** to zoom in and out along the current cursor position.



These settings do not change the behavior of the **Zoom Up** and **Zoom Down** buttons on the **Basic Viewing** and **Standard Viewing** toolbars to zoom in and out along the 3D scene viewpoint center.



### Configuring Display Style

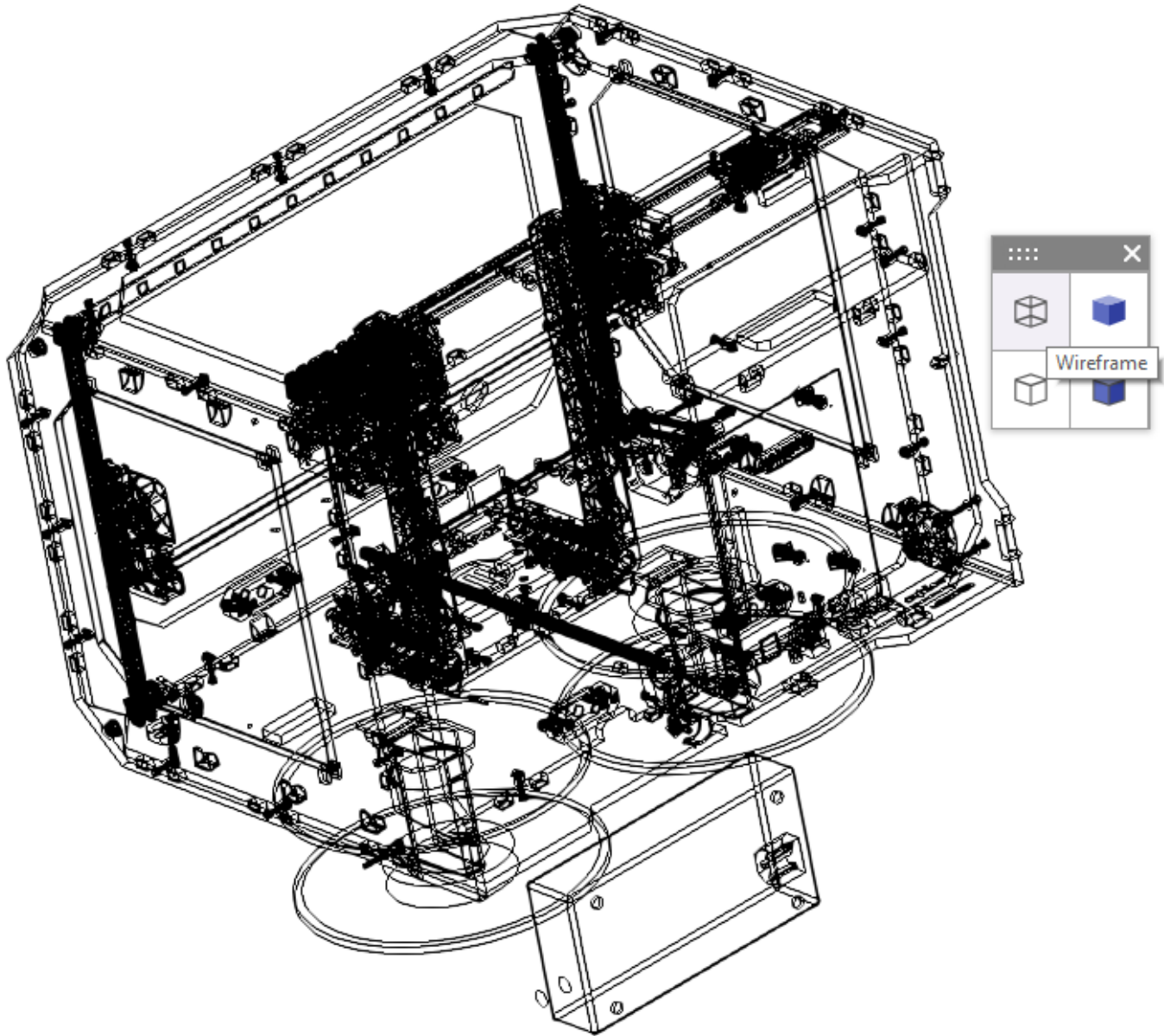
The **Set Display Style** button on the **Standard Viewing** toolbar provides access to a separate toolbar with options on how to display a given 3D CAD model.



With the toolbar in question, an end user can set the following ways of displaying a 3D CAD model:

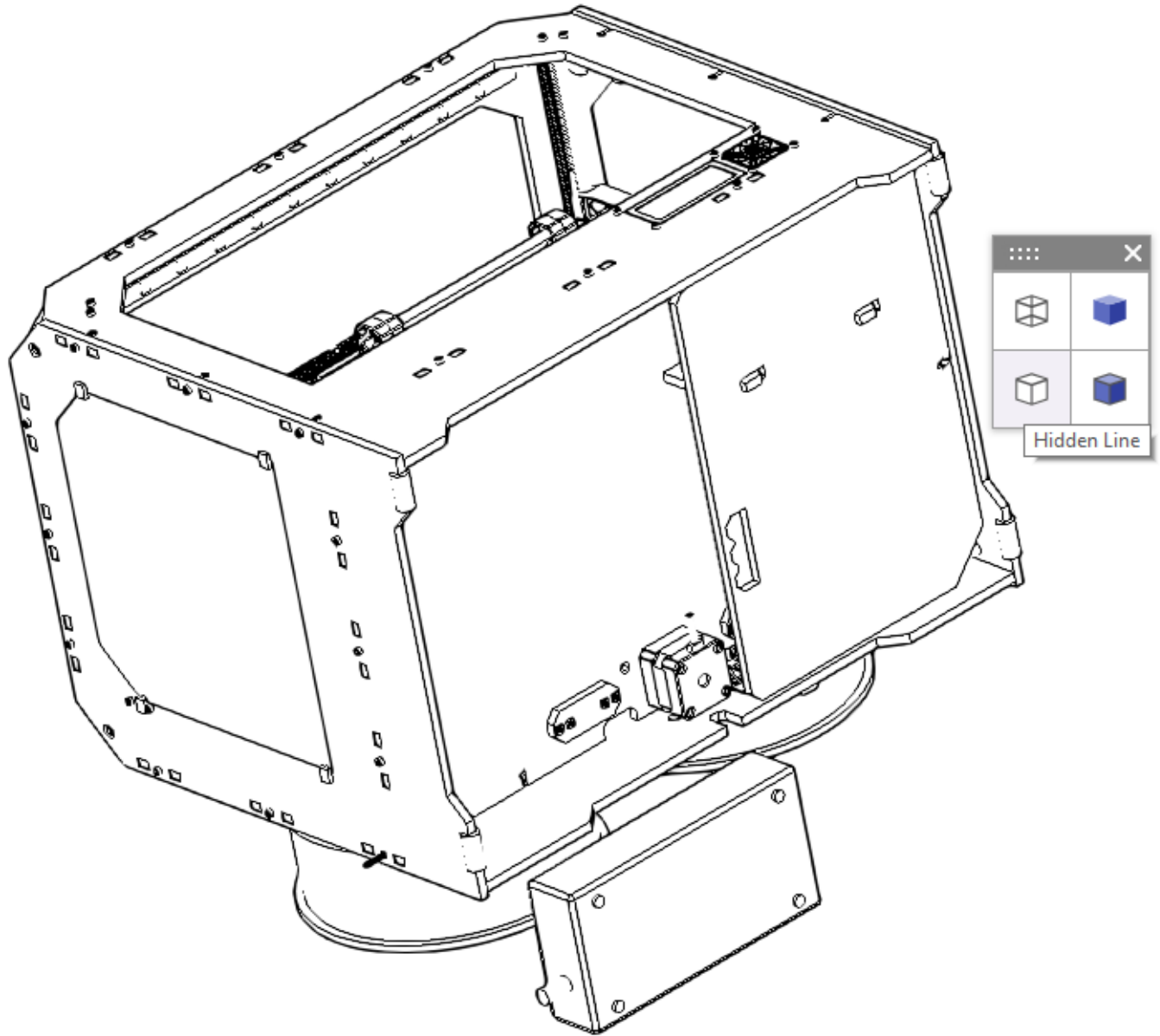


1. **Wireframe**: outer and inner model wireframes (edges) are shown in black. All surfaces are transparent.

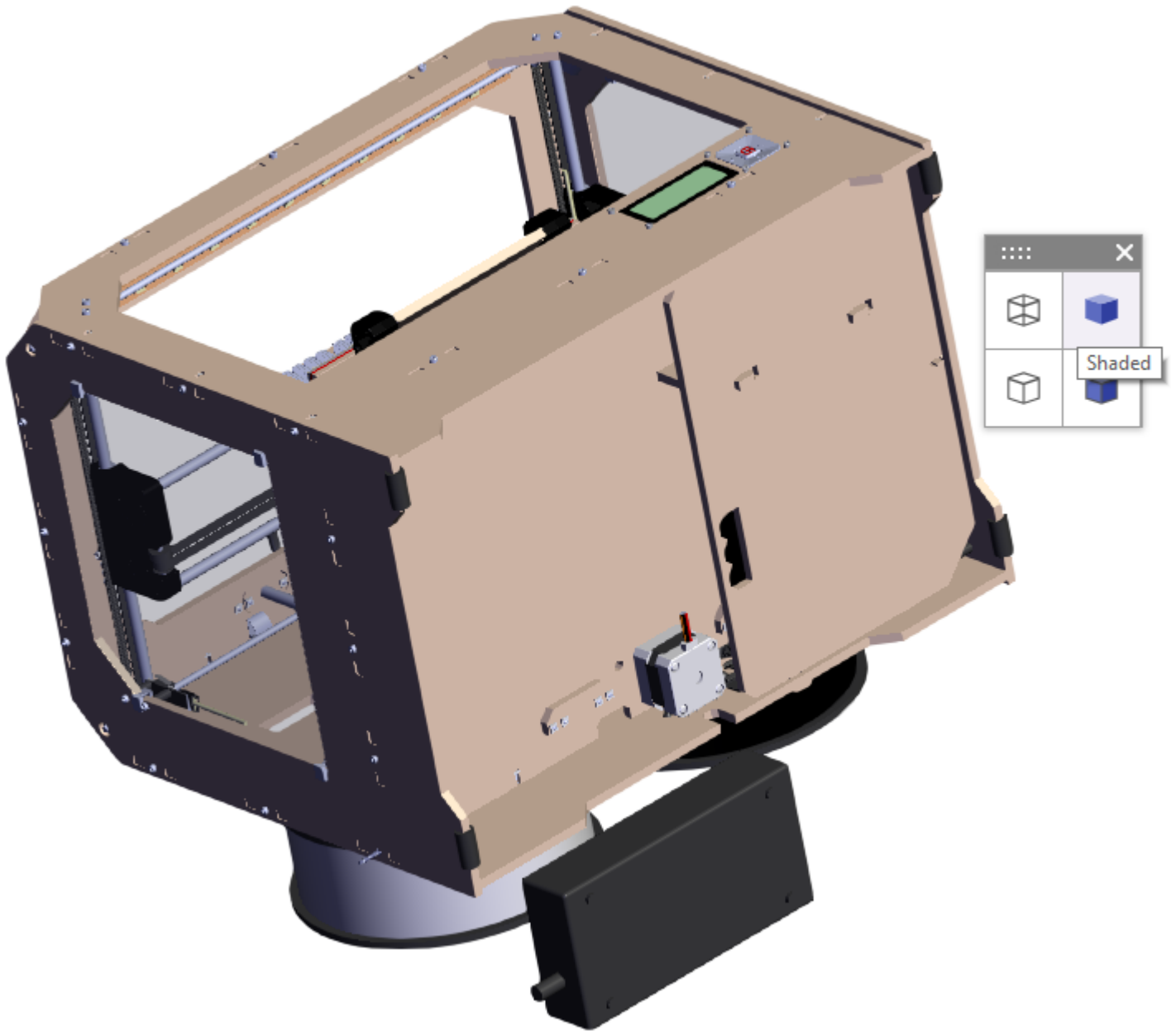


2. **Hidden Line:** outer model surfaces are shown in white with black edges.

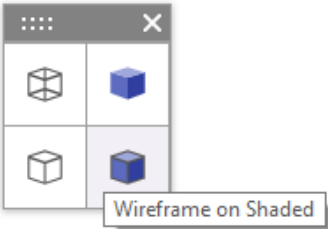
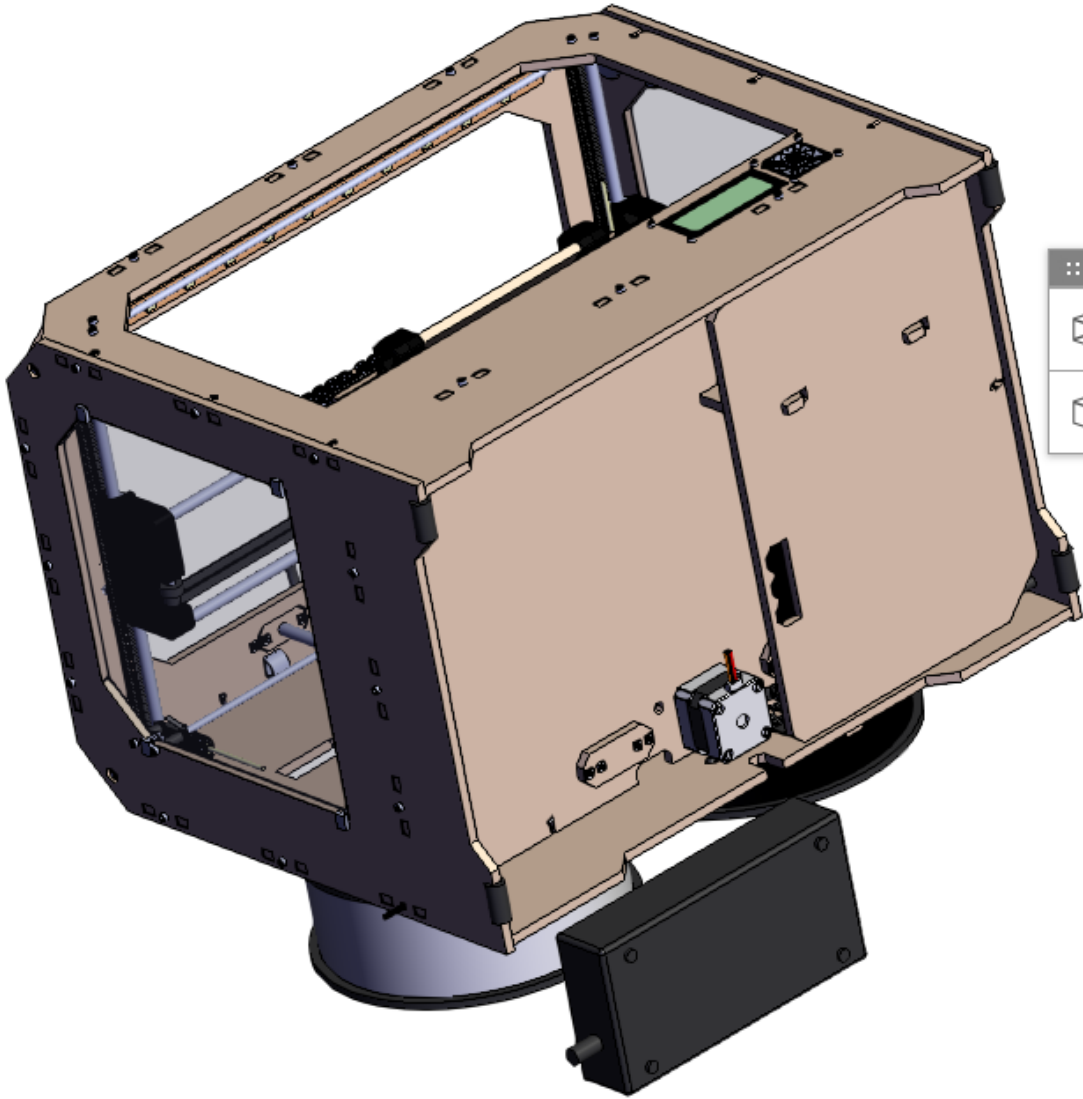




3. **Shaded:** outer model surfaces are shown in color with transparent edges.



4. **Wireframe on Shaded:** outer model surfaces are shown in color with black edges. This is the default display style.



## Measuring CAD Models

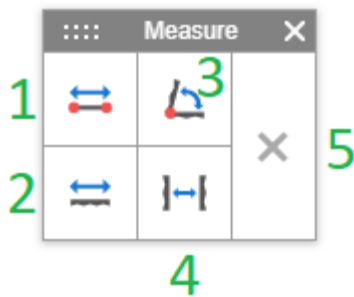
The **Measure** button on the **Standard Viewing** toolbar provides access to the separate **Measure** toolbar with options for measuring the geometrical parameters of a given 3D CAD model.



With the **Measure** toolbar, end users can perform the following measurements of a CAD model:

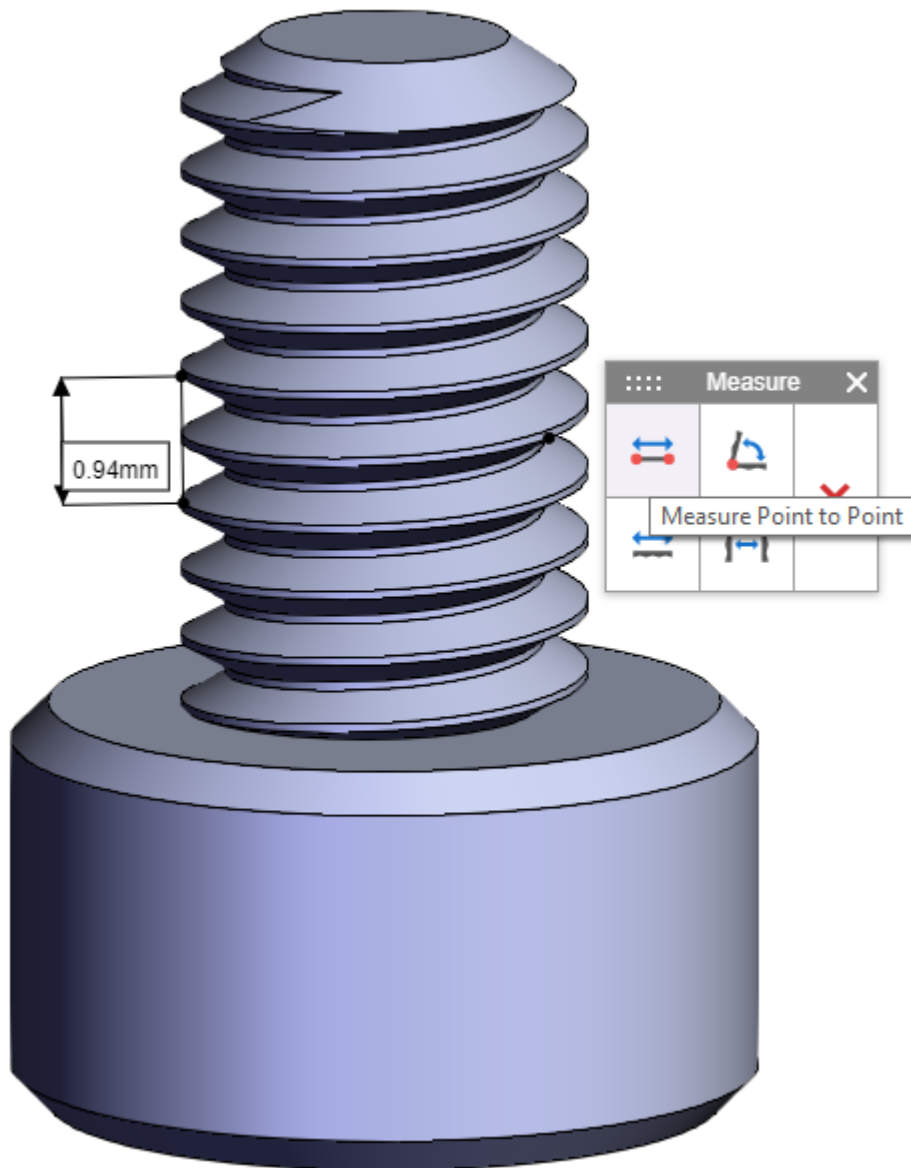
### Important

Note: Measurements are displayed in the CAD model authored units.



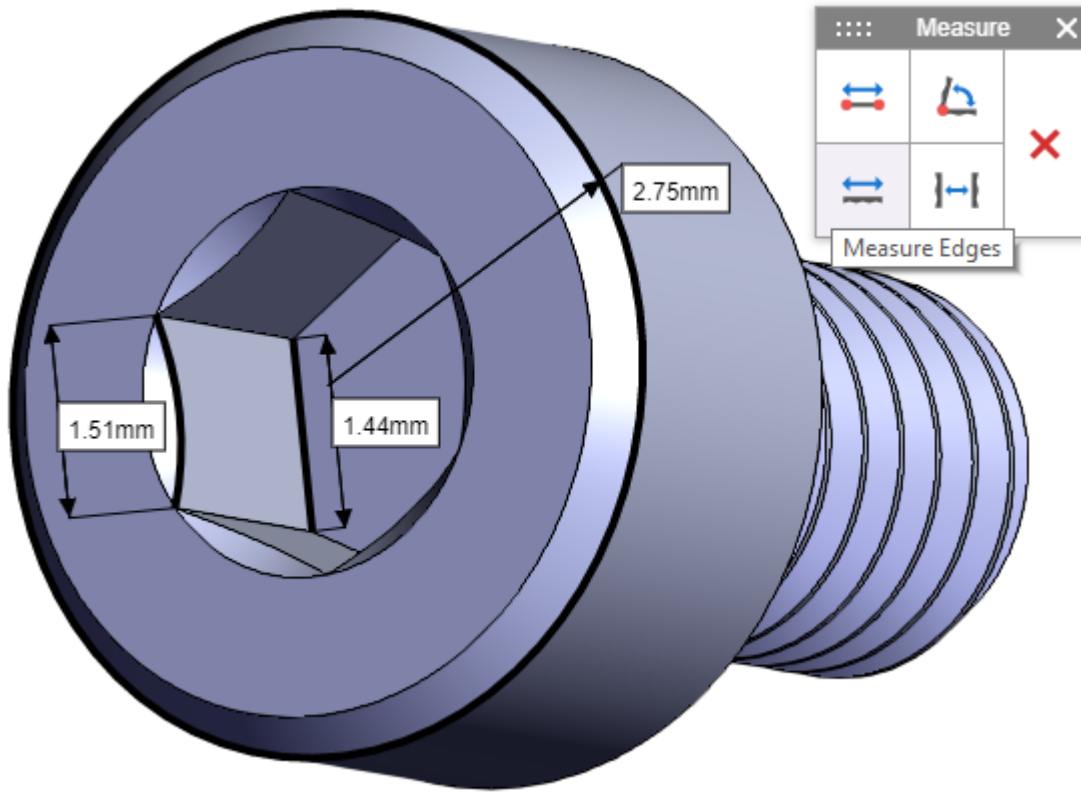
1. **Measure Point to Point:** a distance between two arbitrary points.

Click this button and then click twice on a model to set these points.



2. **Measure Edges**: a length or radius of an arbitrary edge depending on the edge type.

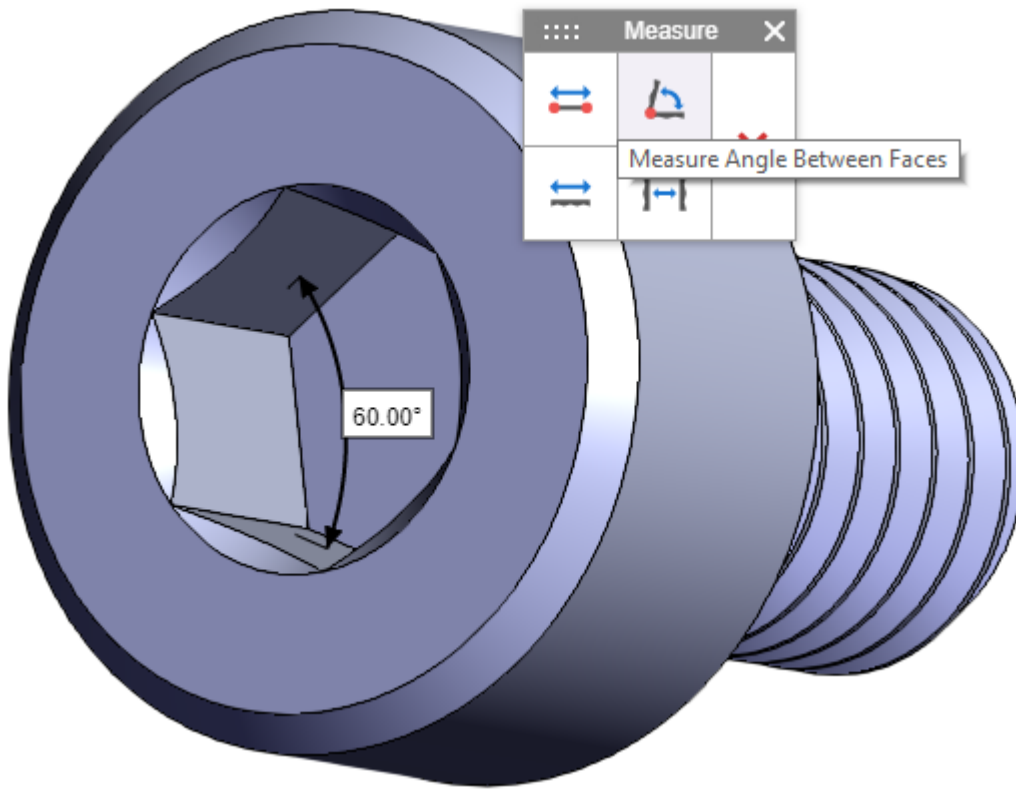
Click this button and then click an edge. The measured edges are bold.



3. **Measure Angle Between Faces:** an angle in grades between two arbitrary surfaces.

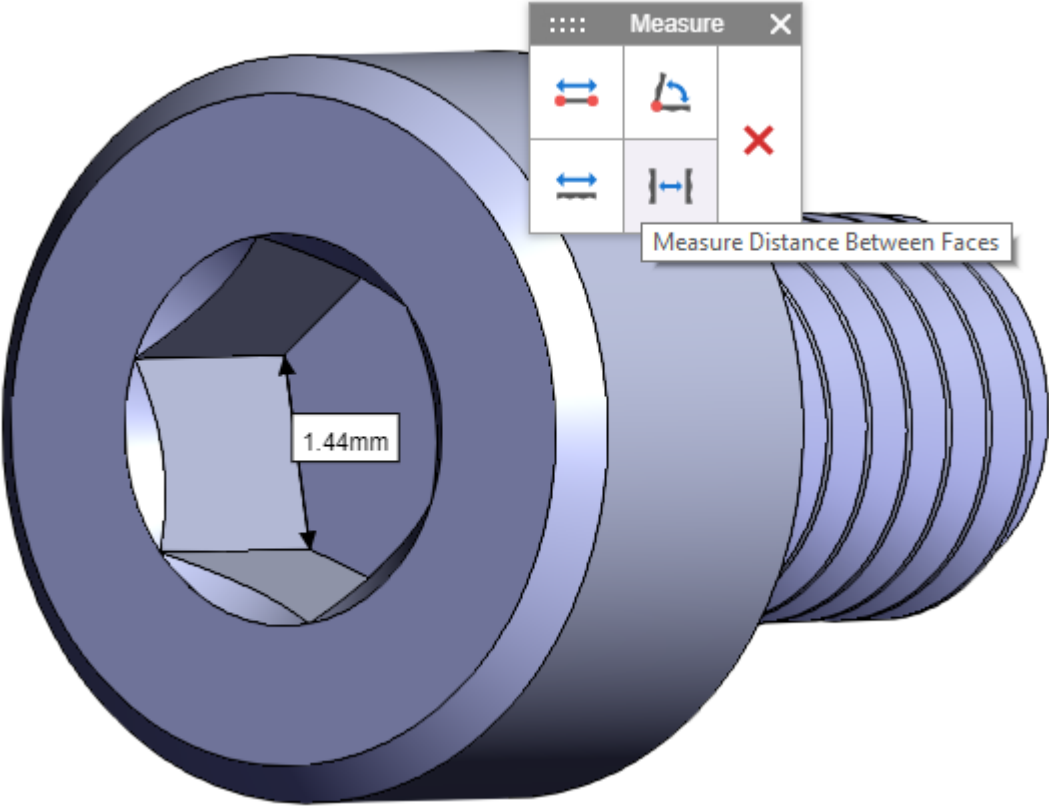
Click this button and then click two appropriate surfaces.



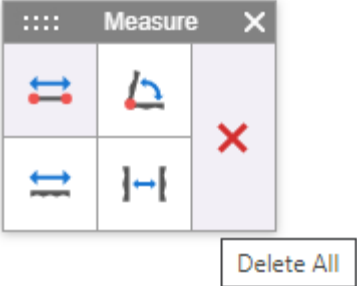


4. **Measure Distance Between Faces:** a distance between two arbitrary surfaces.

Click this button and then click two appropriate surfaces.



5. Click **Delete All** to clear a model from all measurements.

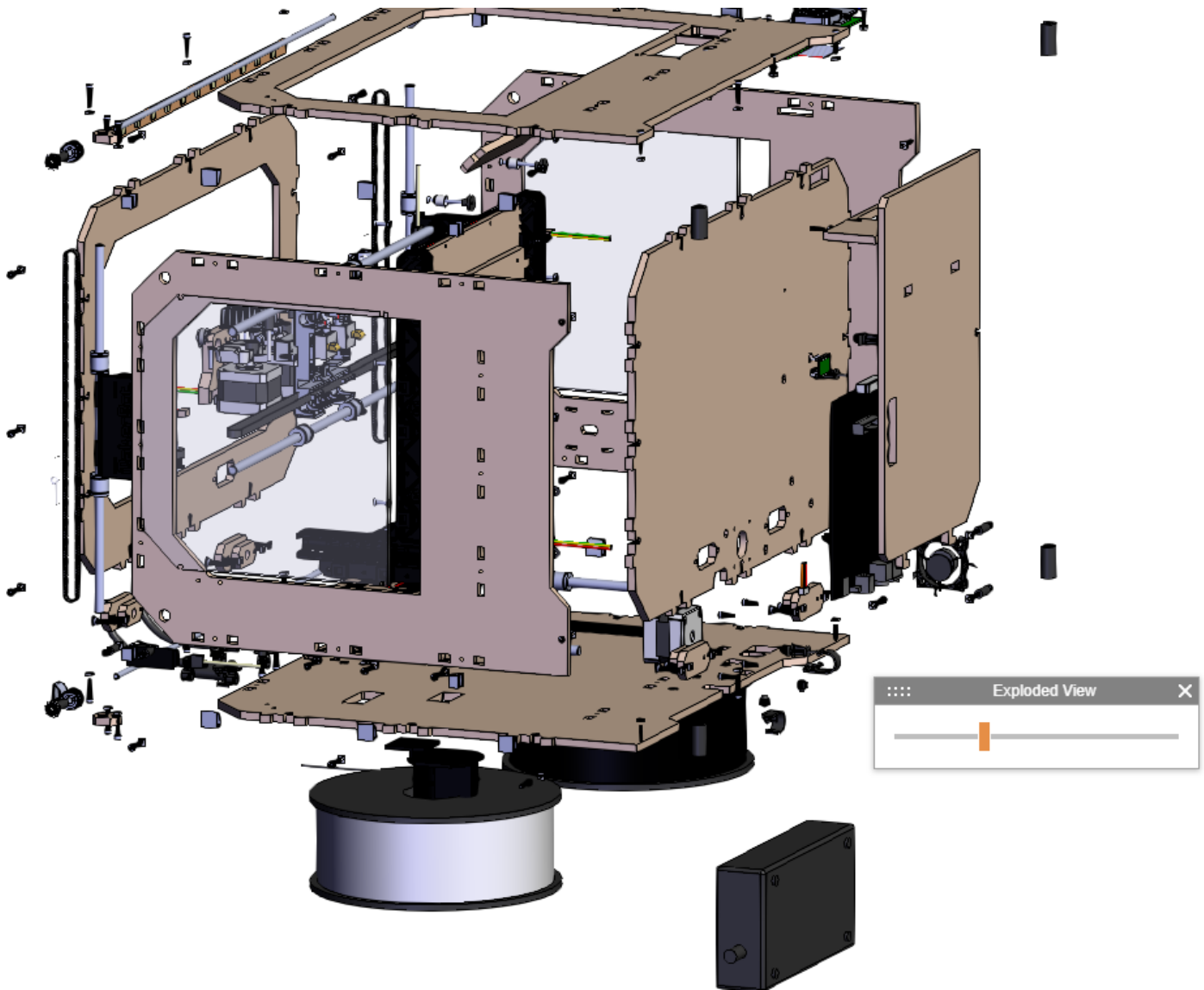


## Exploding CAD Models

The **Exploded View** button on the **Standard Viewing** toolbar provides access to the separate **Exploded View** toolbar.



With the **Exploded View** toolbar, end users can view assemblies as a set of separated parts to be fitted by dragging the slider: the more it is moved to the right, the more separated the parts become.



In the exploded view, other viewing and markup features are available except for measurements. A measurement that had been done before exploding a given 3D model is hidden during exploding and becomes visible once this 3D model is shown not exploded.

To quit the exploded view, either:

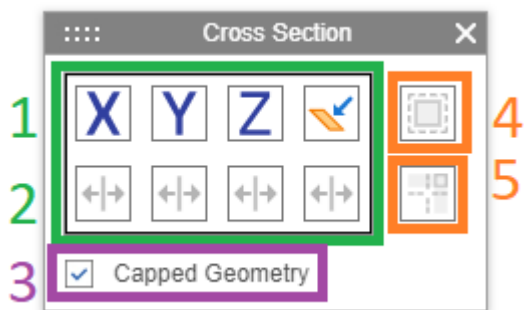
- Close the **Exploded View** toolbar
- Click **Reset View** in the **3D Scene** or **Part** context menu.

### Cross Section Viewing of the CAD Model

The **Cross Section** button on the **Standard Viewing** toolbar provides access to the separate **Cross Section** toolbar.

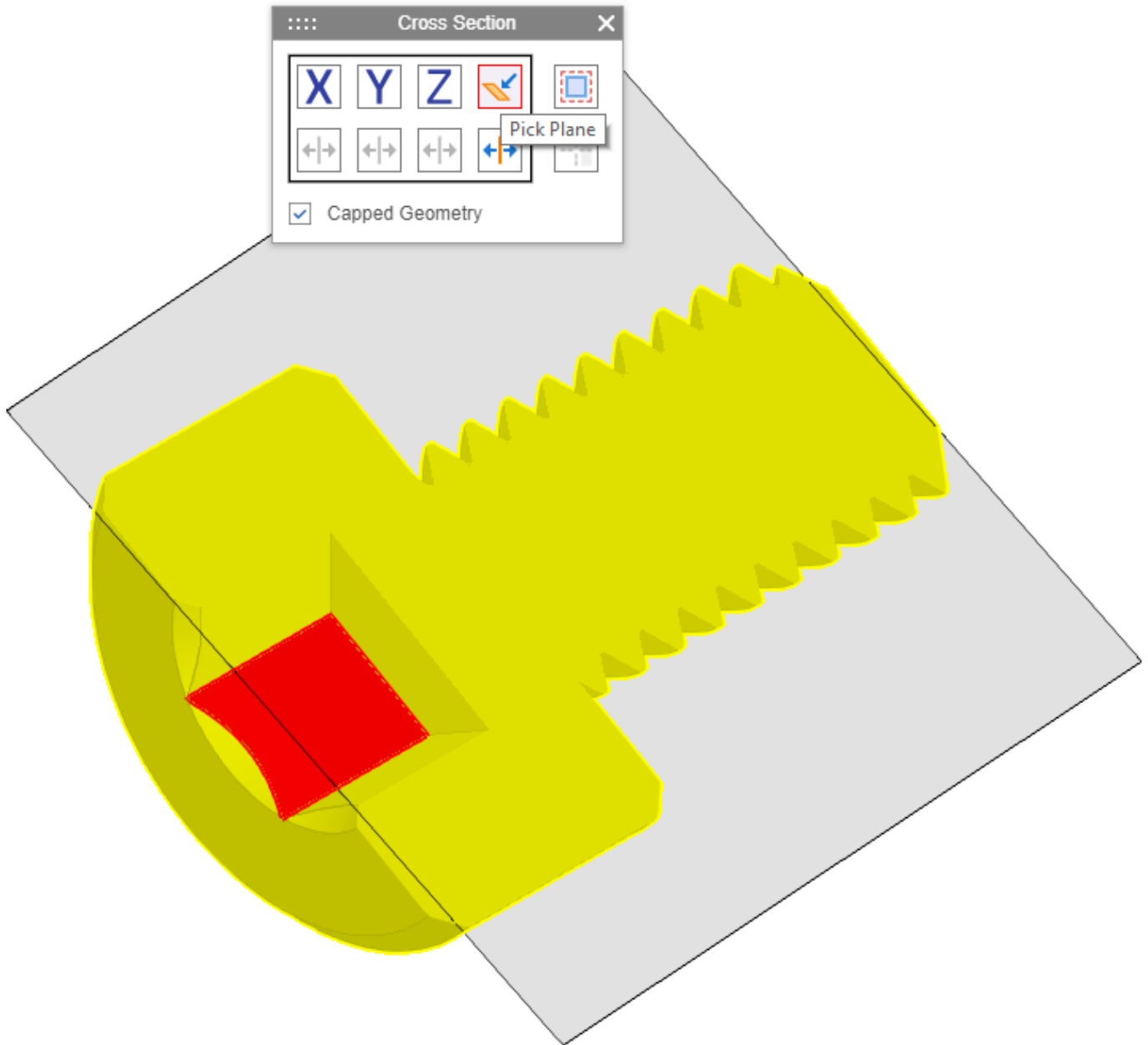


With the **Cross Section** toolbar, end users can view a 3D CAD model in sections as follows:

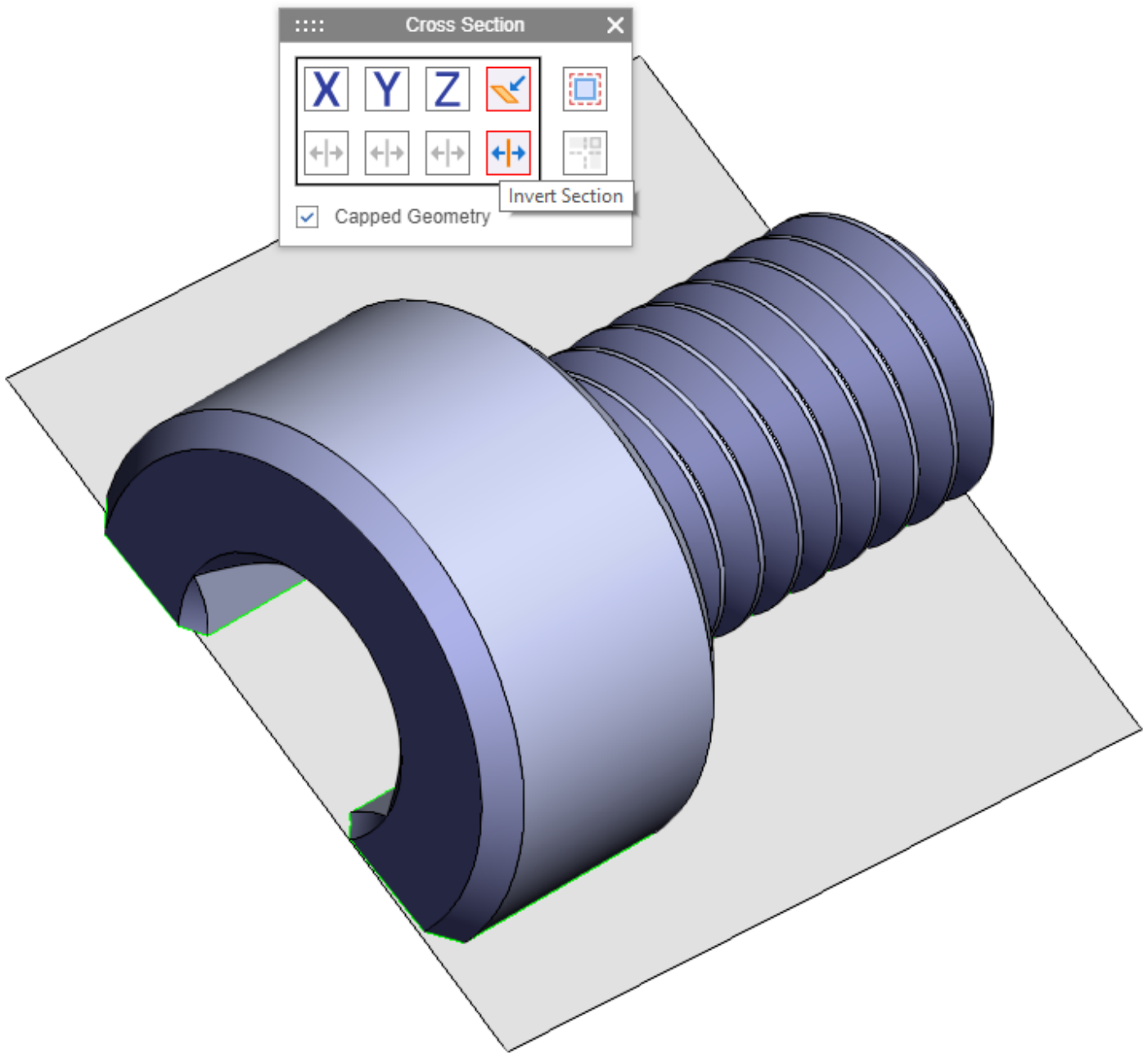


1. Model cutting plates can be specified with the set of plane buttons:
  - o The **X**, **Y**, and **Z** buttons are to set the cutting planes according to the model coordinate system.
  - o The **Pick Plane** button is to set an arbitrary cutting plane by clicking a target surface and then clicking this button.

Dragging the cutting planes cuts the model into sections.

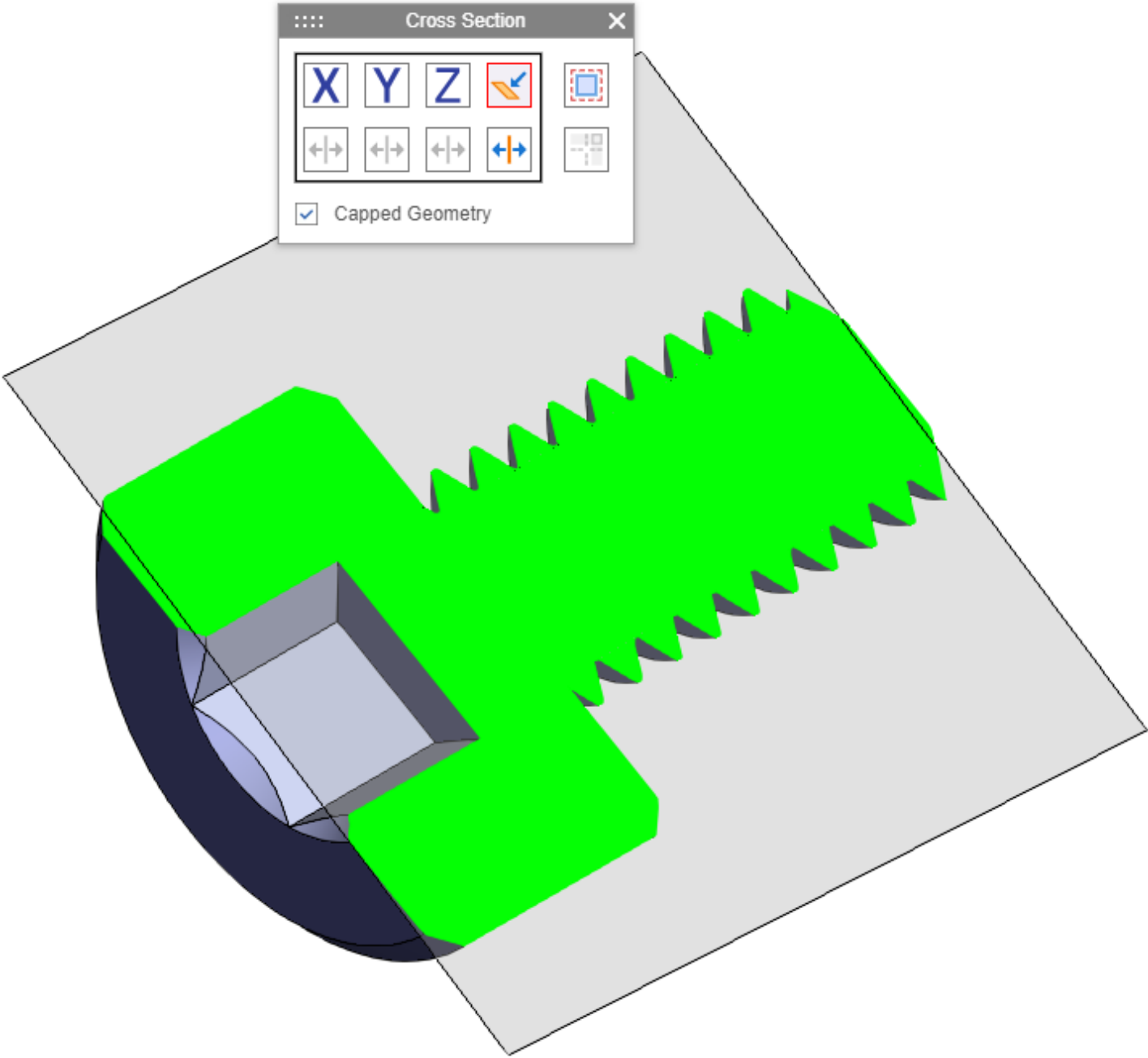


2. Each plane button includes its own **Invert Section** button to determine which of the cut sections is visible.



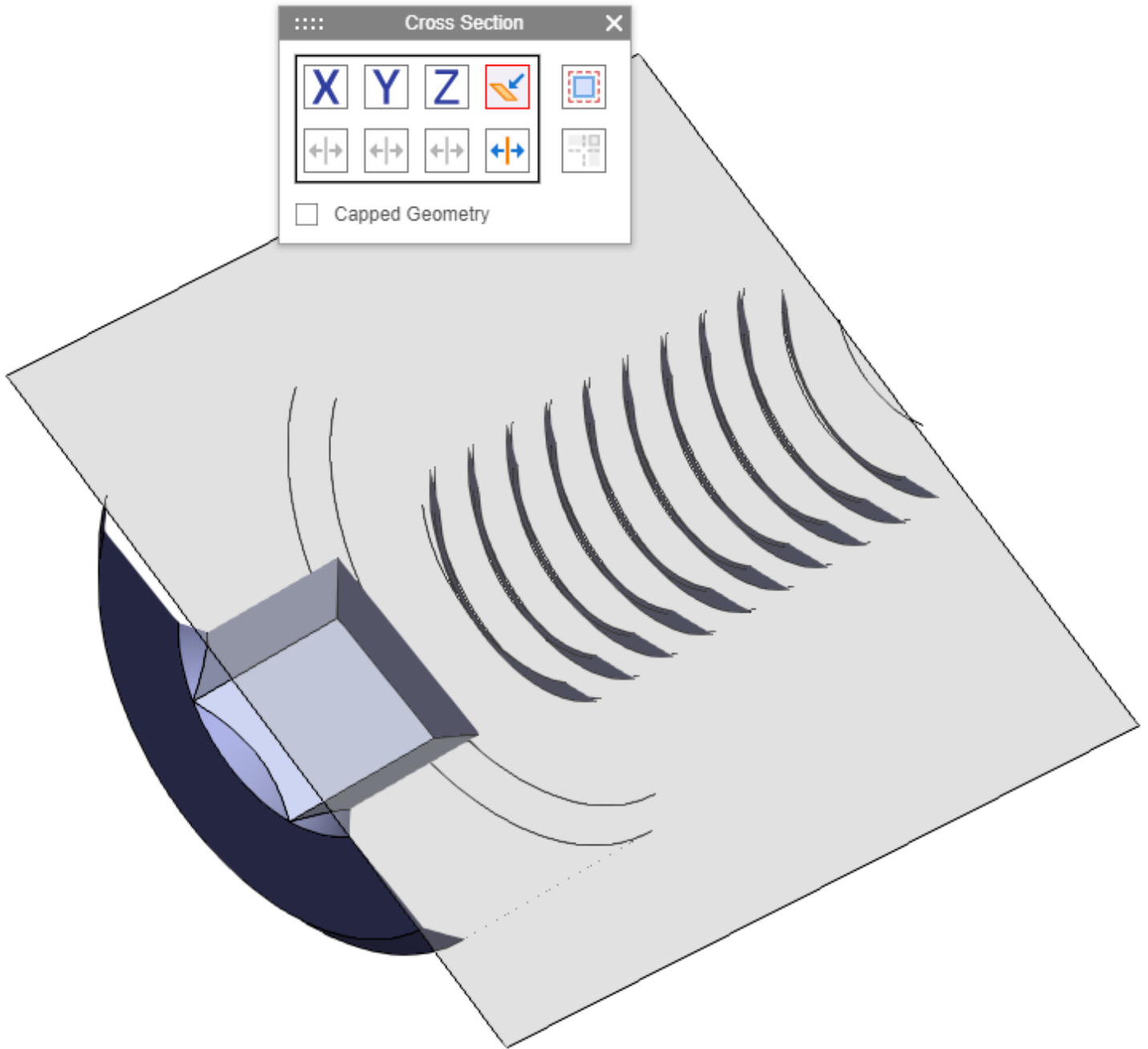
3. Selecting the **Capped Geometry** check box displays a cut section in green.





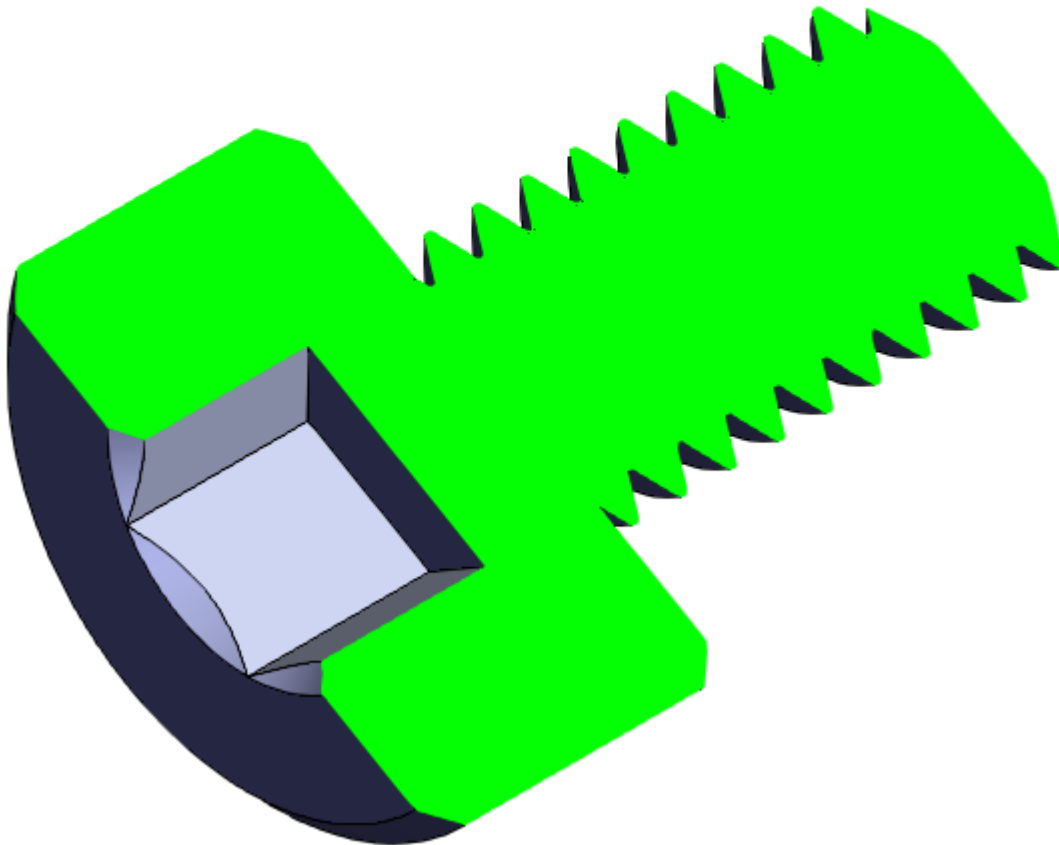
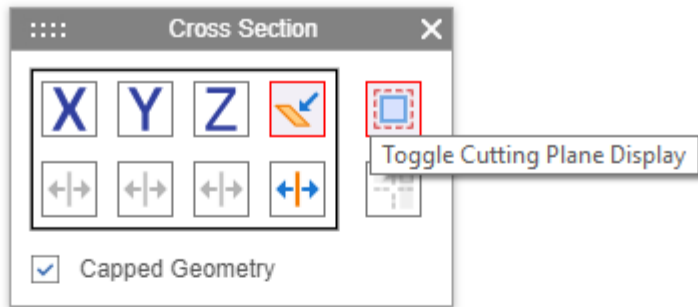
If this check box is cleared, a cut section is hollow.



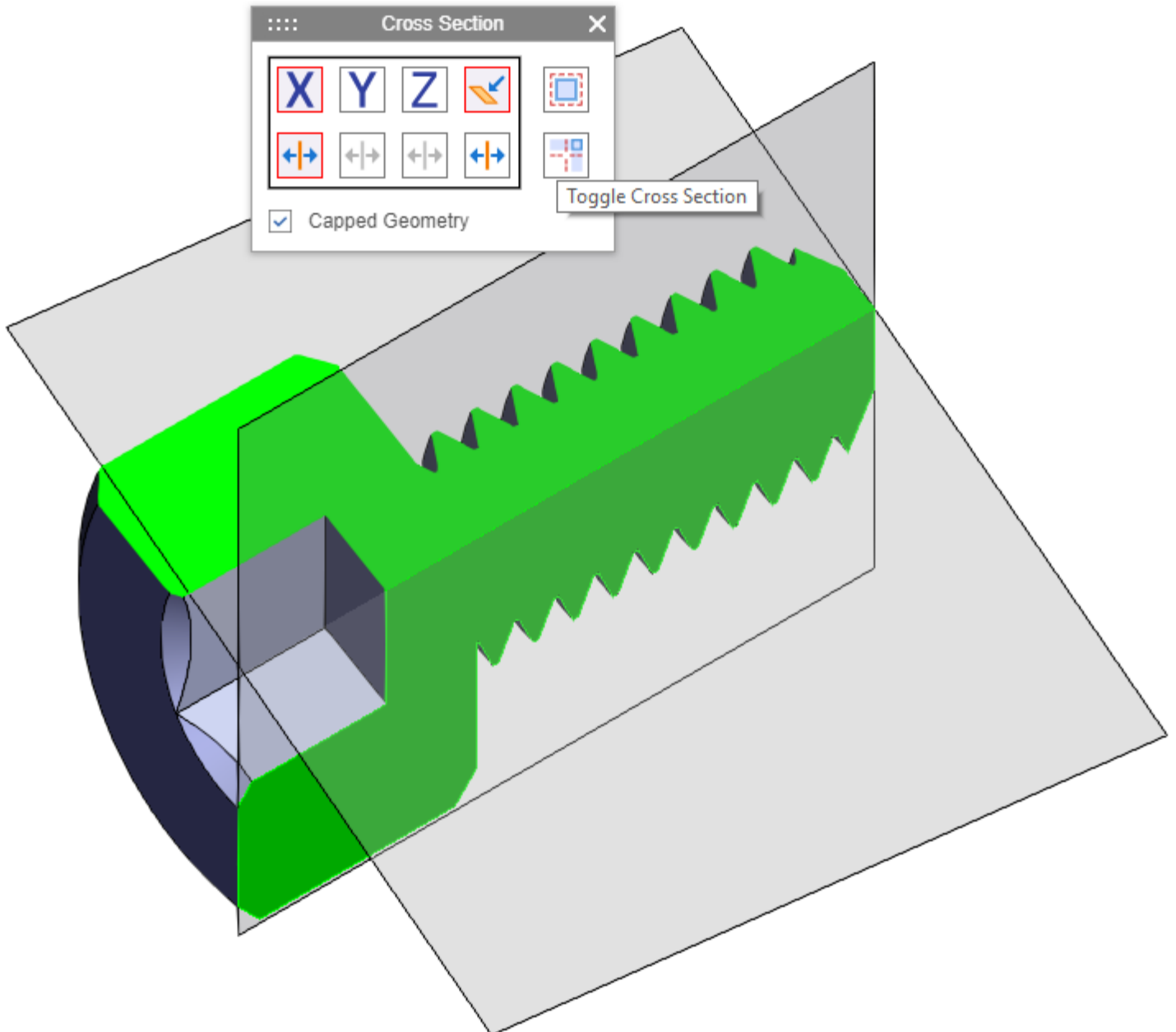


4. Clicking the **Toggle Cutting Plane Display** button hides or shows the cutting planes.

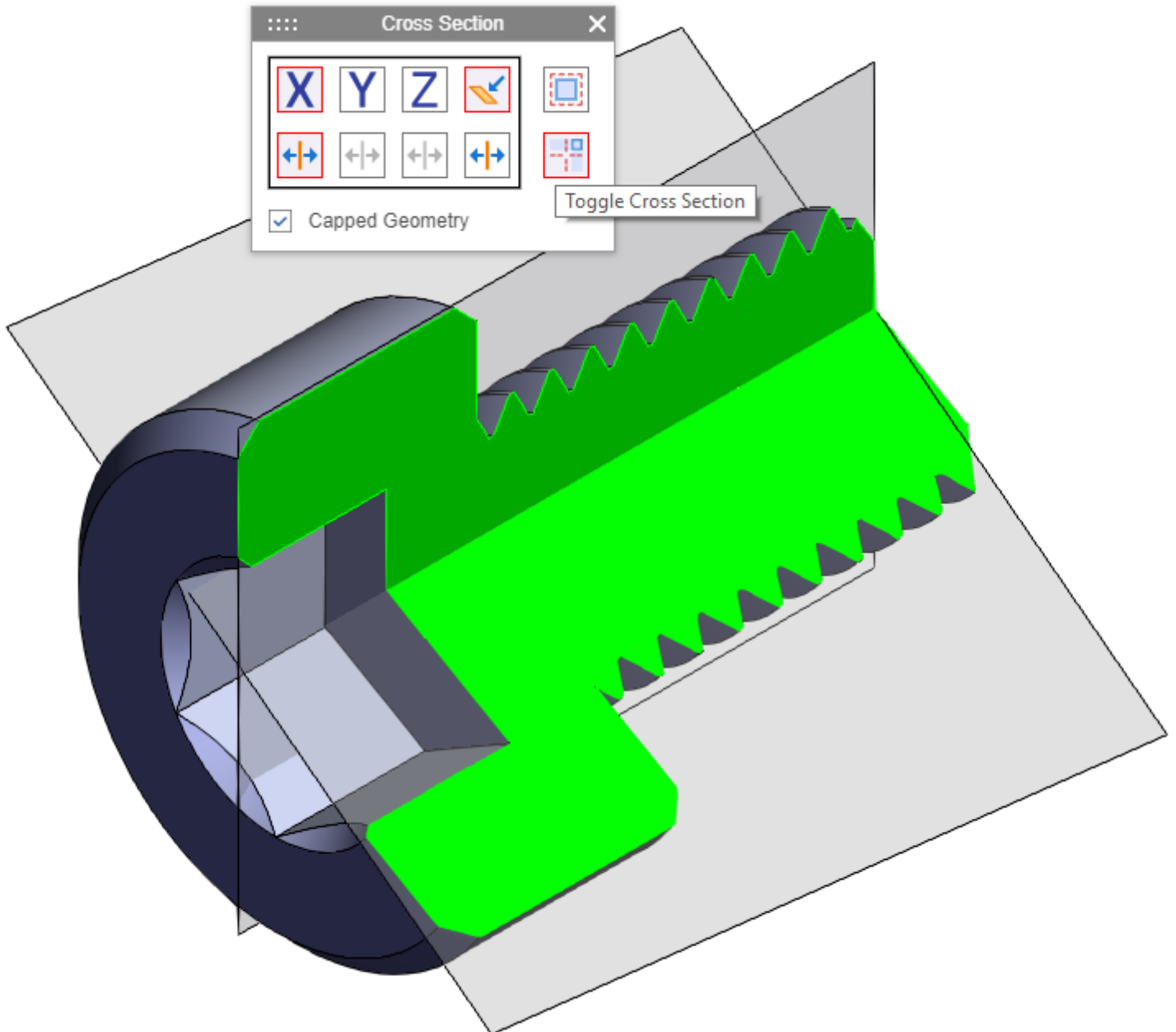




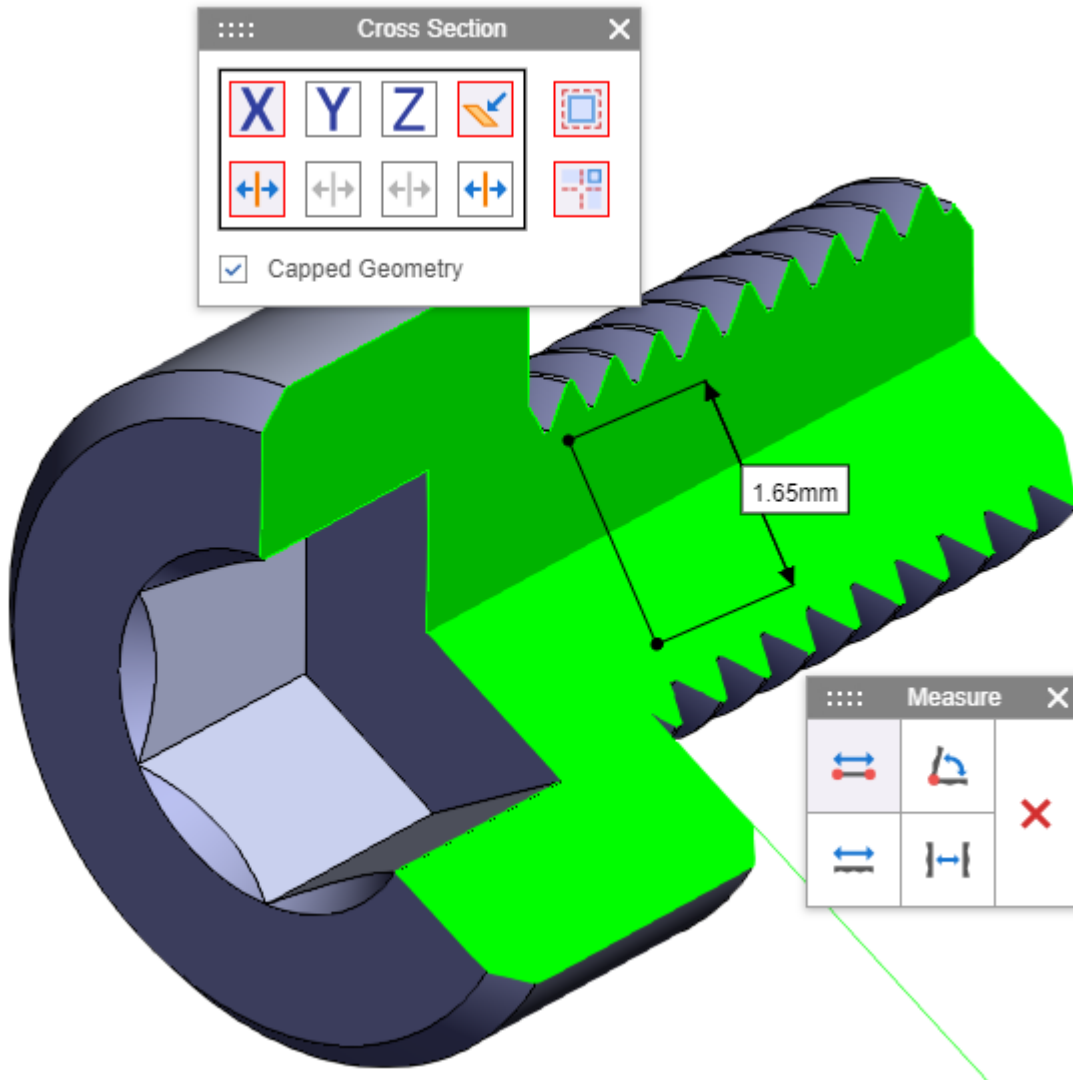
5. When using several cutting planes, the **Toggle Cross Section** button becomes available for selecting which sides of the planes can cut.



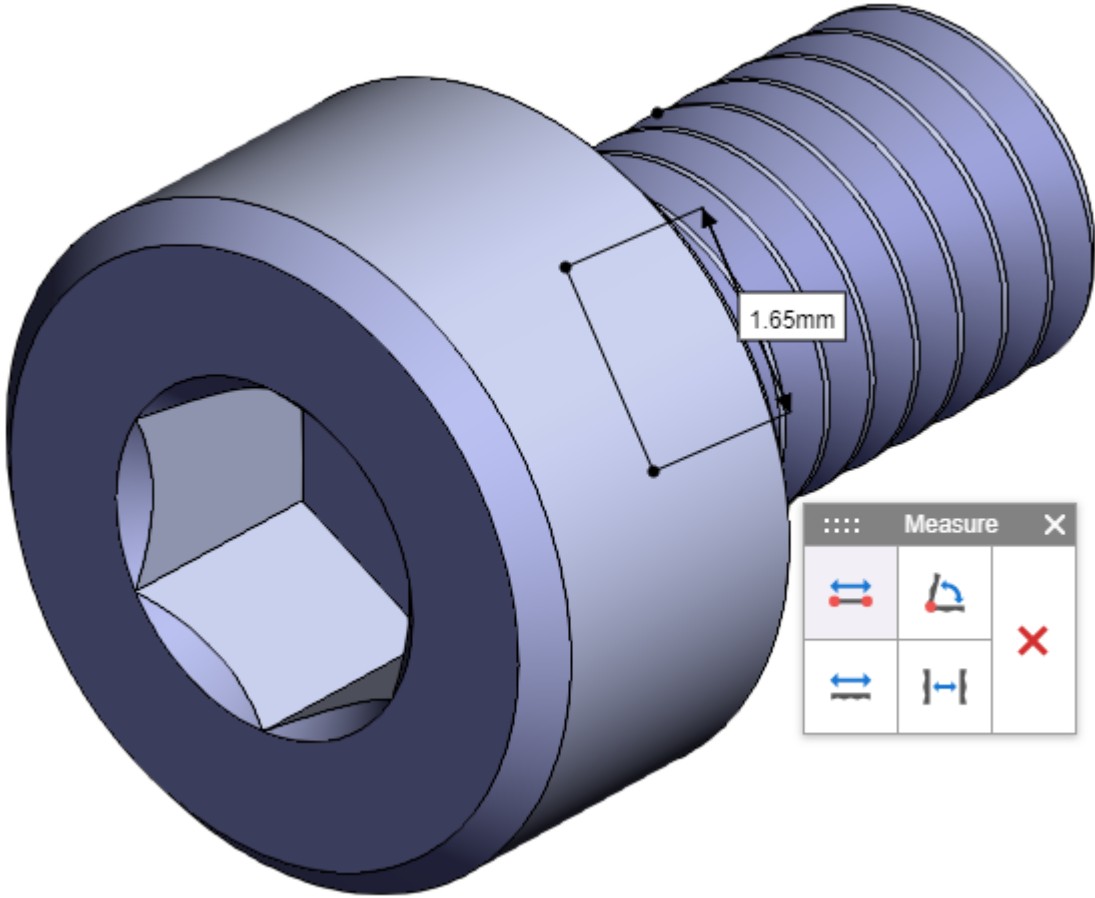
Clicking the **Toggle Cross Section** button toggles the cutting sides of the planes. This button can be combined with the **Invert Section** buttons of these planes.



To enable other viewing and markup functionality in the cross-section model view, the cutting planes should be hidden.



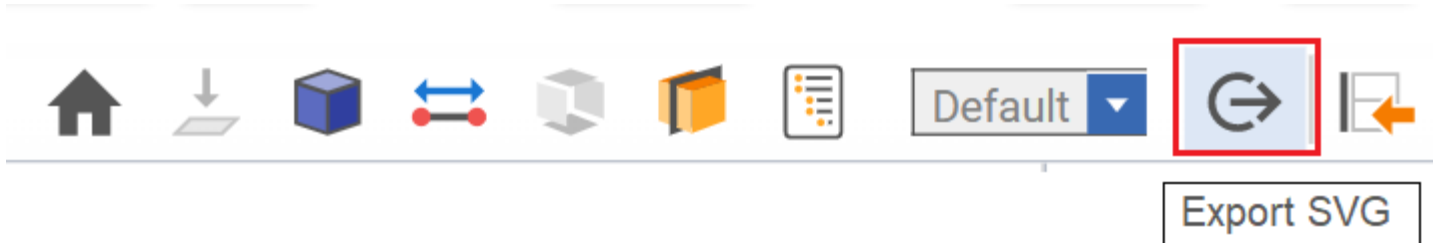
To quit the cross-section model view, the **Cross Section** toolbar should be closed.



## Exporting SVG Image File

The following steps outline the process of exporting the SVG image file of a current 3D View to a local machine using **Export SVG**:

1. Select a necessary Dynamic View Definition.
2. From the **StandardViewing** toolbar, click **Export SVG**.



The **Save SVG Image** dialog box appears.

### Save Svg Image



Save

Cancel

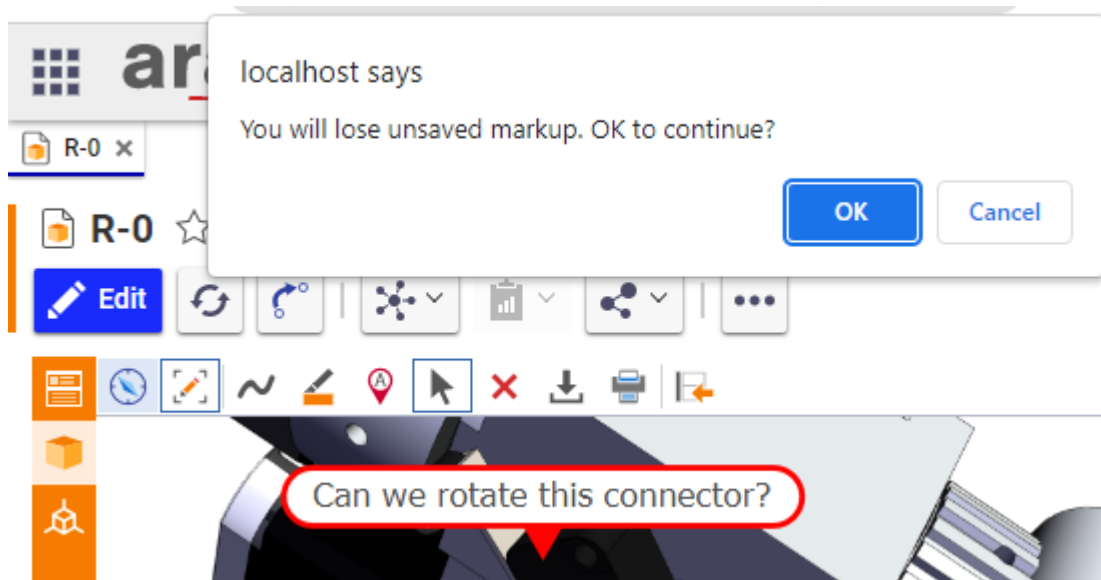
3. Click **Save**.



## Marking up CAD Models

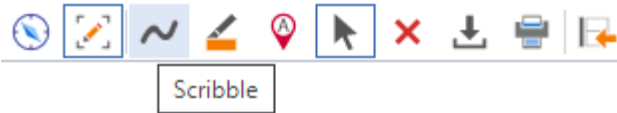
In the 3D model markup mode launched with the **Markup** button on the **Basic Markup** or **Standard Markup** toolbar, an end user can annotate a 3D CAD model with arbitrary lines, highlighted areas, and labels.

A 3D model view with markup items should be explicitly saved before switching to the model viewing mode because all current markups are not saved and will be lost during this switching: the viewing 3D scene will not have them.

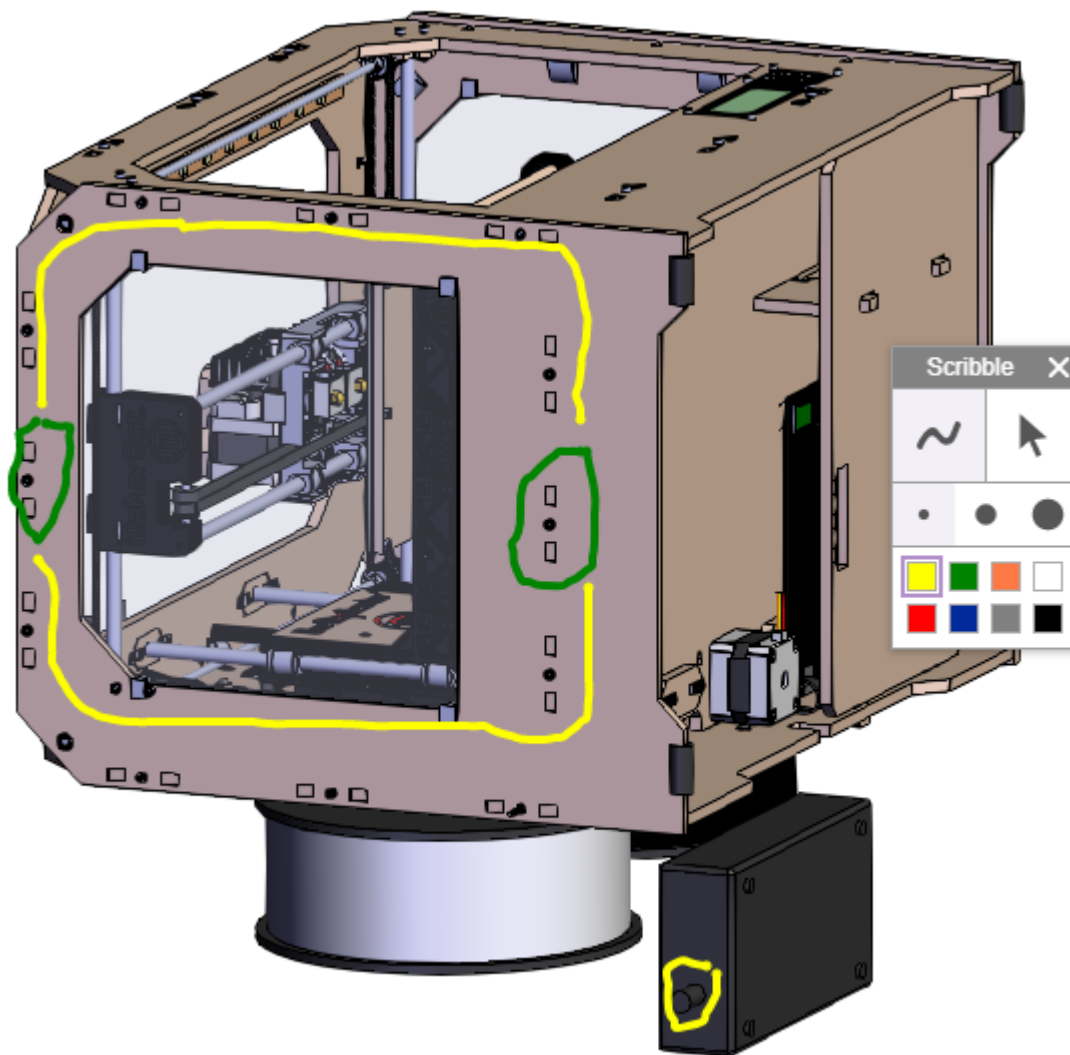


### Drawing on CAD Models

The **Scribble** button on the **Basic Markup** or **Standard Markup** toolbar provides access to the scribble feature.

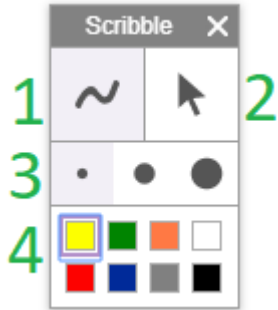


With the discussed feature, an end user can draw arbitrary lines on a 3D CAD model.

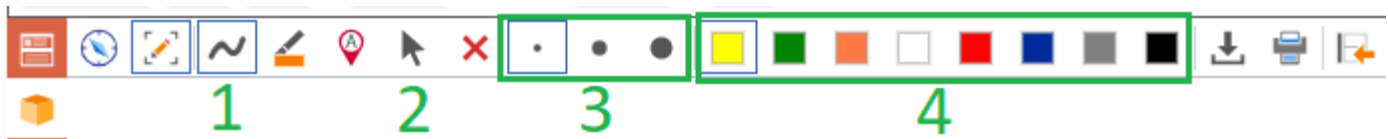


Depending on a given toolbar, the feature is displayed as follows:

- **Basic Markup:** as the separate **Scribble** toolbar.



- **Standard Markup:** as additional buttons on this toolbar.



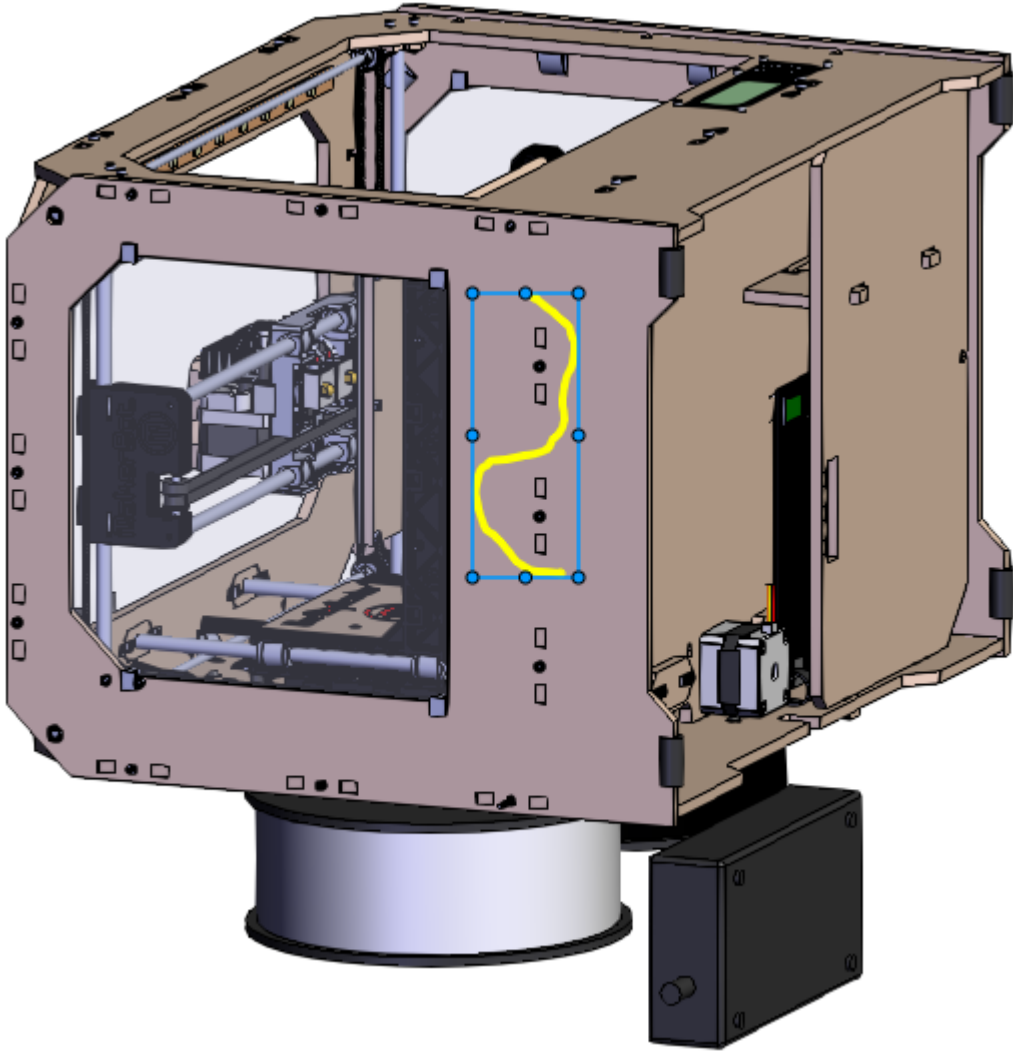
The scribble feature provides the following functionality:

1. **Scribble:** to draw arbitrary lines on a 3D CAD model.
2. **Select:** to select markups on a 3D CAD model.
3. **Line widths:** to set the width of a line to be drawn.
4. **Line colors:** to set the color of a line to be drawn.

To use a feature or option, a user should click its button.

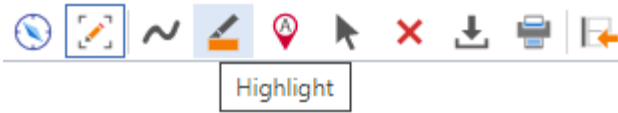
To select a line, a user should click **Select** and then select this line. A selected line has blue edges with circles and cannot be repositioned.



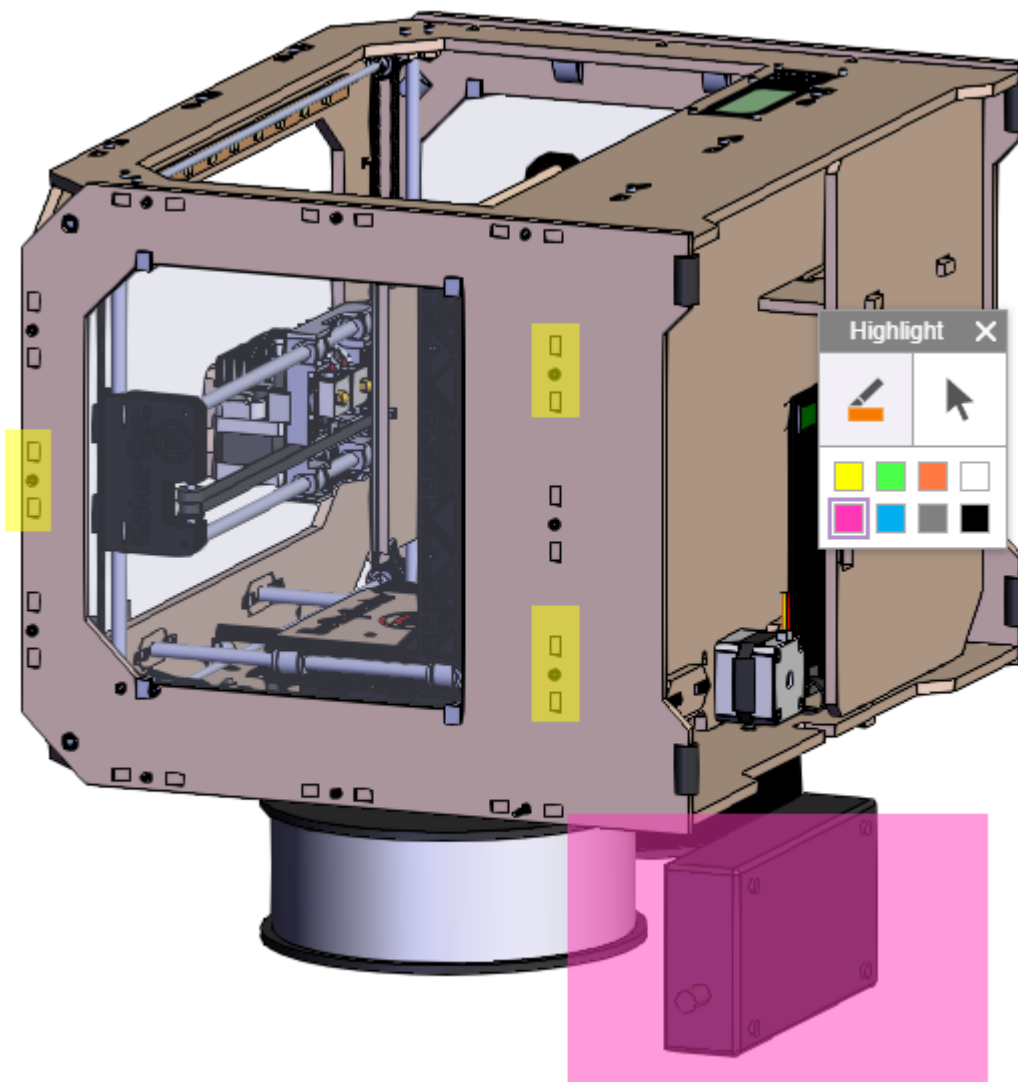


### Highlighting Areas on CAD Models

The **Highlight** button on the **Basic Markup** or **Standard Markup** toolbar provides access to the highlighting feature.

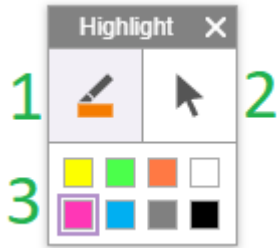


With the discussed feature, an end user can draw arbitrary rectangular areas on a 3D CAD model.



Depending on a given toolbar, the feature is displayed as follows:

- **Basic Markup:** as the separate **Highlight** toolbar.



- **Standard Markup:** as additional buttons on this toolbar.



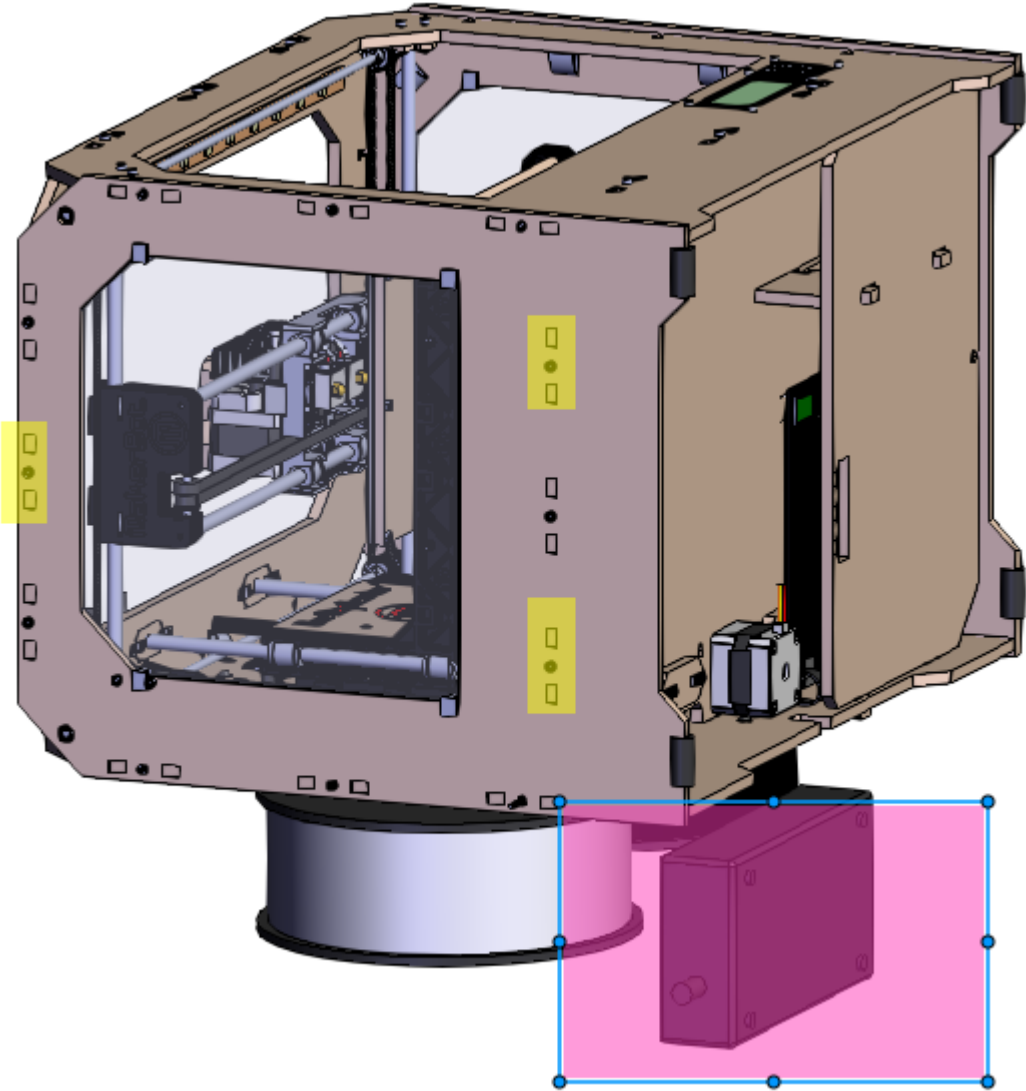
The highlighting feature provides the following functionality:

1. **Highlight:** to draw arbitrary rectangular areas on a 3D CAD model.
2. **Select:** to select markups on a 3D CAD model.
3. **Rectangle colors:** to set the color of a rectangular area to be drawn.

To use a feature or option, a user should click the required button.

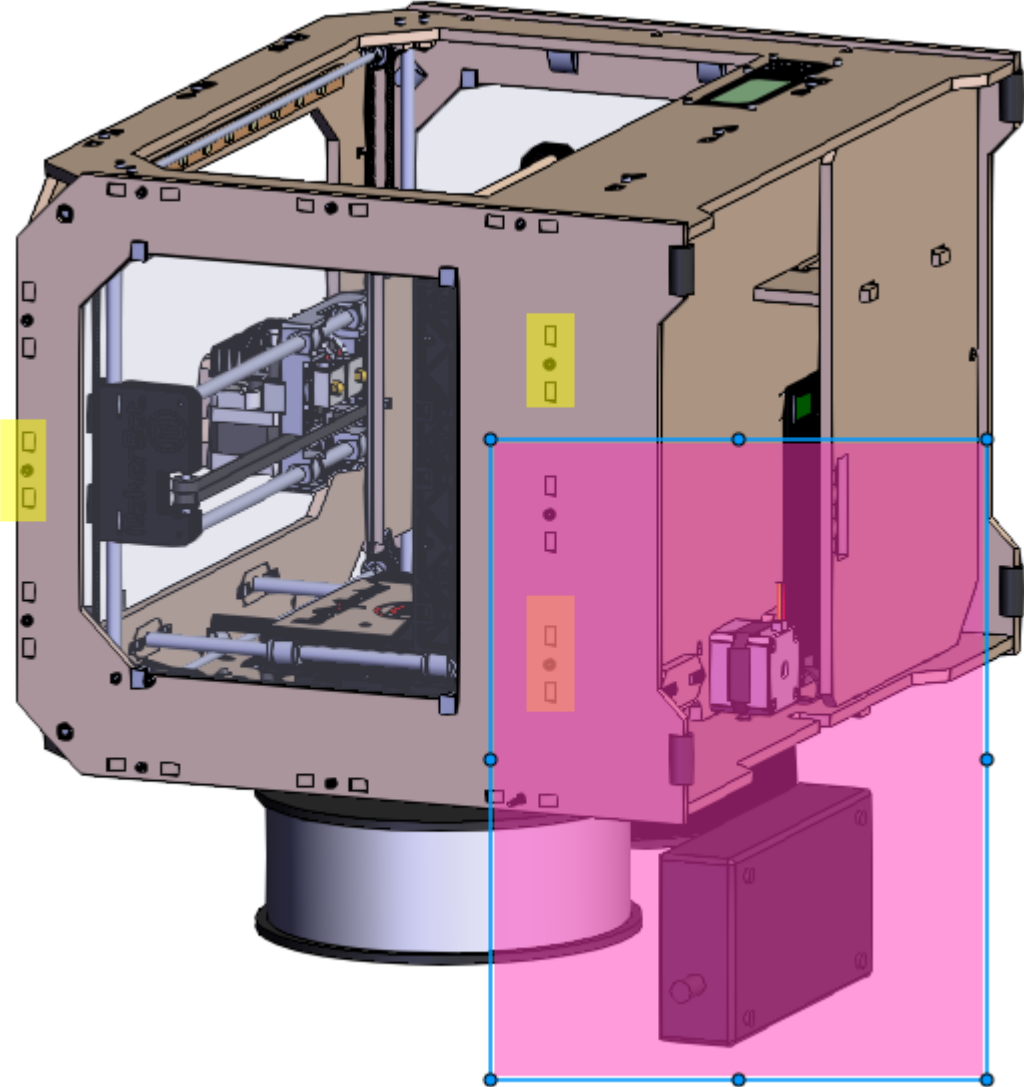
To select a drawn rectangle, a user should click **Select** and then select this rectangle. A selected rectangle has blue edges with circles and can be repositioned.





Dragging the blue circles on the edges of a selected rectangle repositions this rectangle.





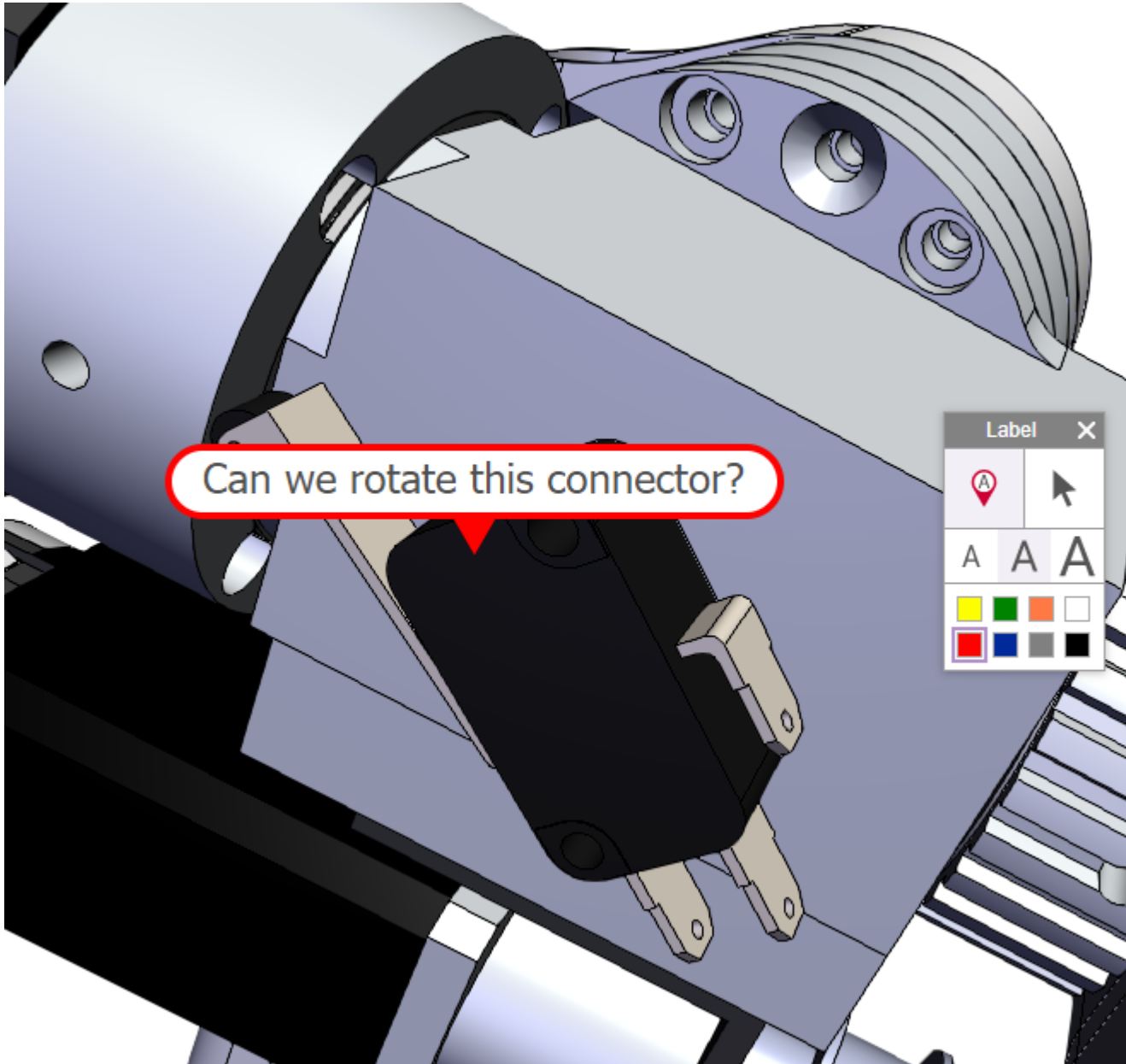
### Putting Labels on CAD Models

The **Label** button on the **Basic Markup** or **Standard Markup** toolbar provides access to the label feature.



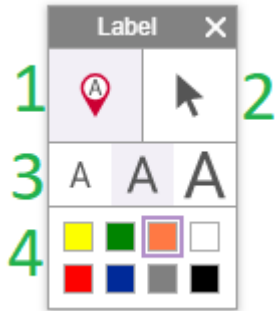
With the discussed feature, an end user can put arbitrary labels with comments on a 3D CAD model.





Depending on a given toolbar, the feature is displayed as follows:

- **Basic Markup:** as the separate **Label** toolbar.



- **Standard Markup:** as **additional** buttons on this toolbar.



The label feature provides the following functionality:

1. **Label:** to add labels to a 3D CAD model.
2. **Select:** to select markups on a 3D CAD model.
3. **Font sizes:** to set the size of a text to be written in a new label.
4. **Label colors:** to set the color of a label to be added.

To use a feature or option, a user should click the required button.

The following steps outline the process to create a label:

1. Click the **Label** button.
2. Click on a 3D CAD model where the label should be put.

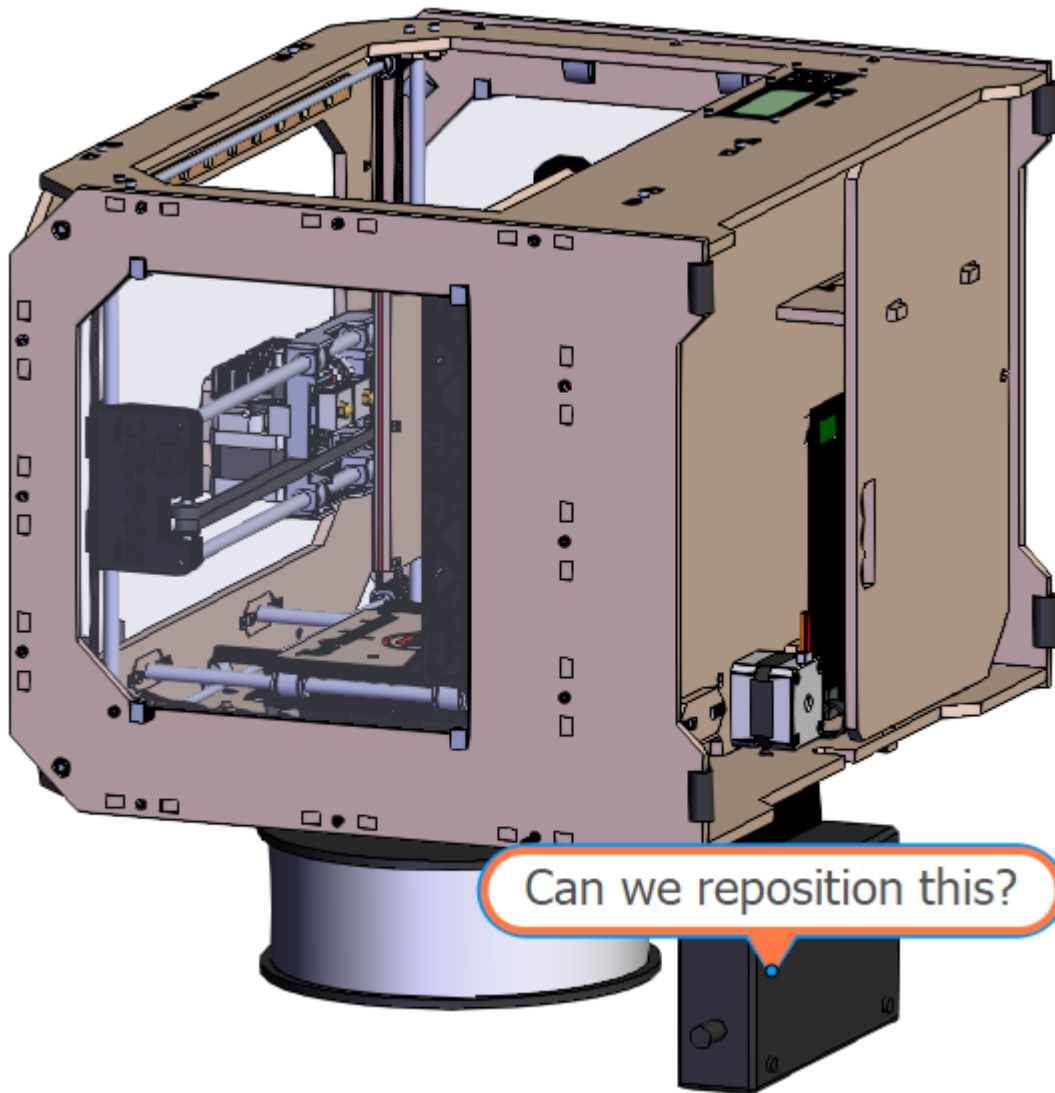


3. Enter the appropriate text inside the label.

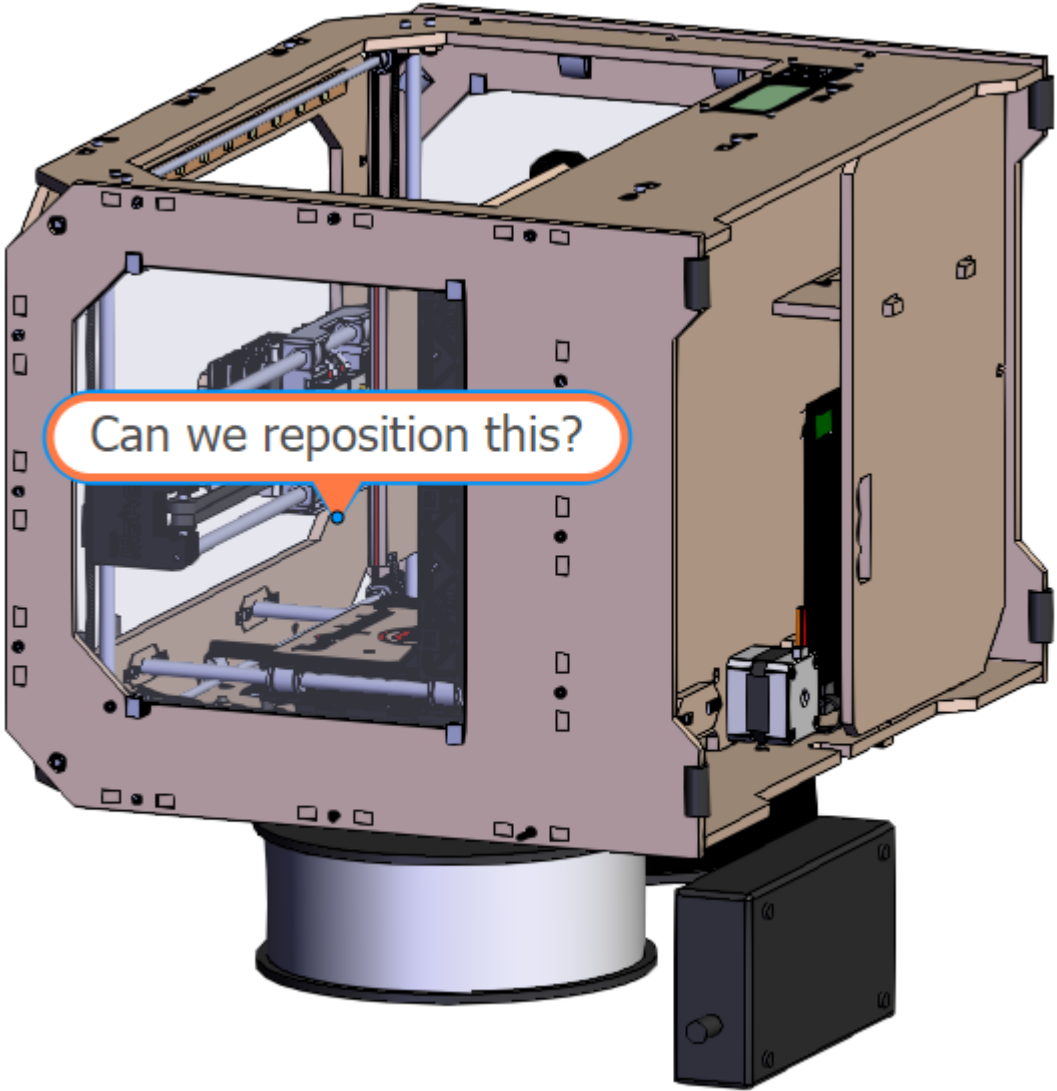




4. Click the **Select** button.
5. To select a label, a user should click **Select** and then select this label. A selected label has blue edges with a circle at the bottom point and can be repositioned.

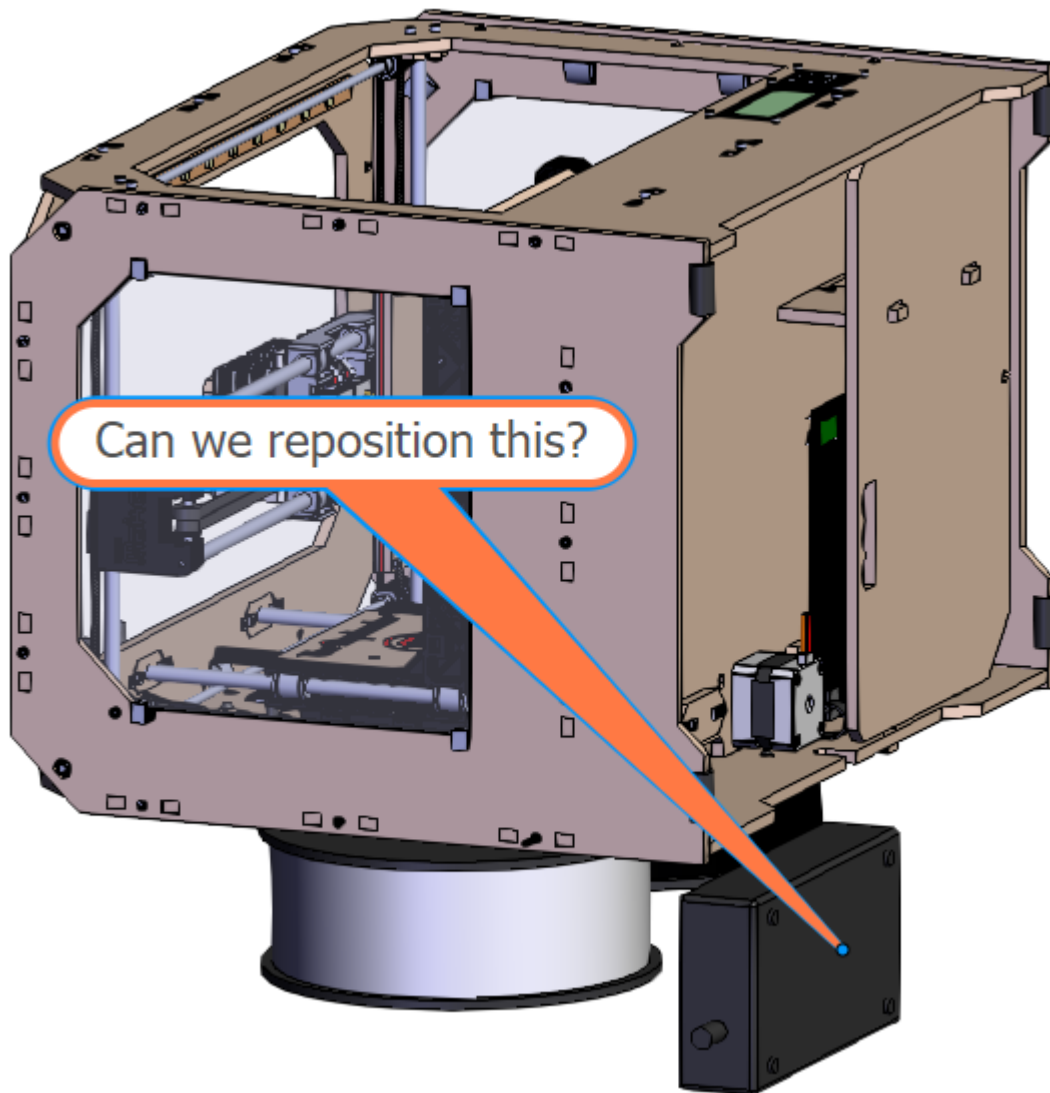


6. To move an existing label to a new position, a user should select this label and drag its colored frame below the text box and above the blue bottom circle.

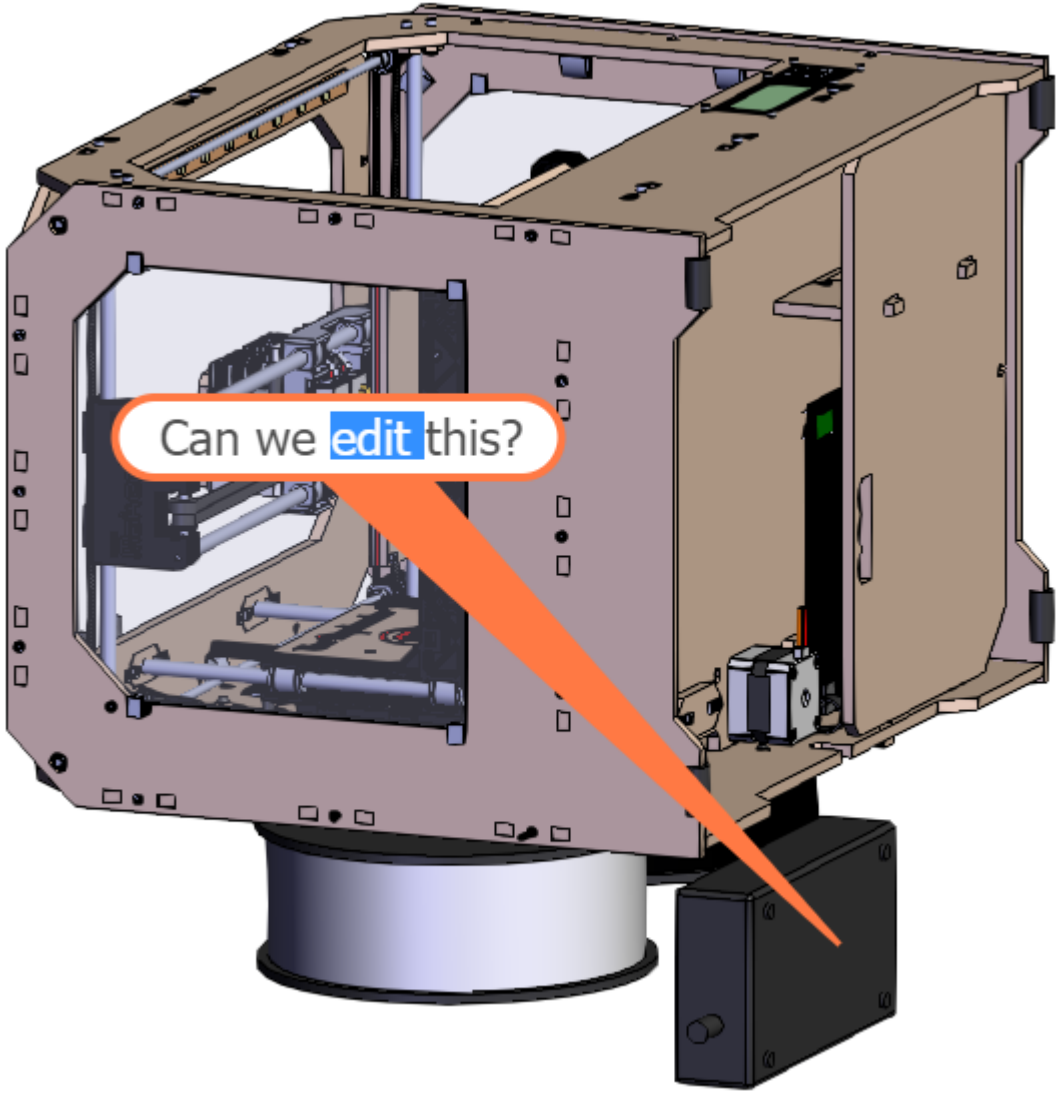


7. To change where a label points to, a user should select this label and drag its blue bottom circle.





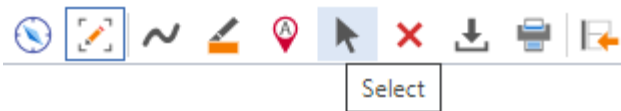
8. To change the label text, a user should click inside the label and edit the text.



### Selecting Markups on CAD Models

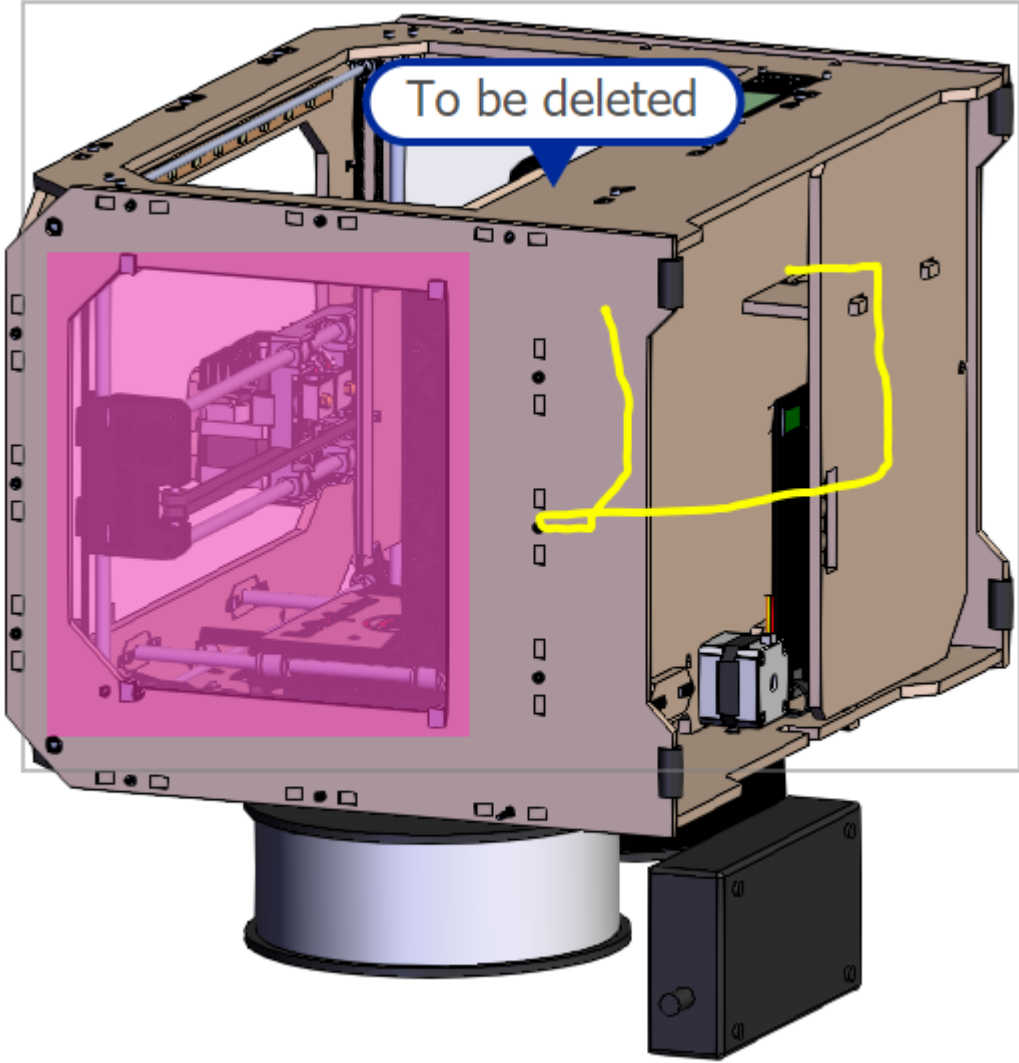
End users can select a single or several markup items for further actions. Selected items have bold blue edges with circles. Further actions, position, and a number of the circles depend on the markup type of each item—see an appropriate section.

To select a single markup item, a user should click **Select** on the **Basic Markup** or **Standard Markup** toolbar and then click the item.



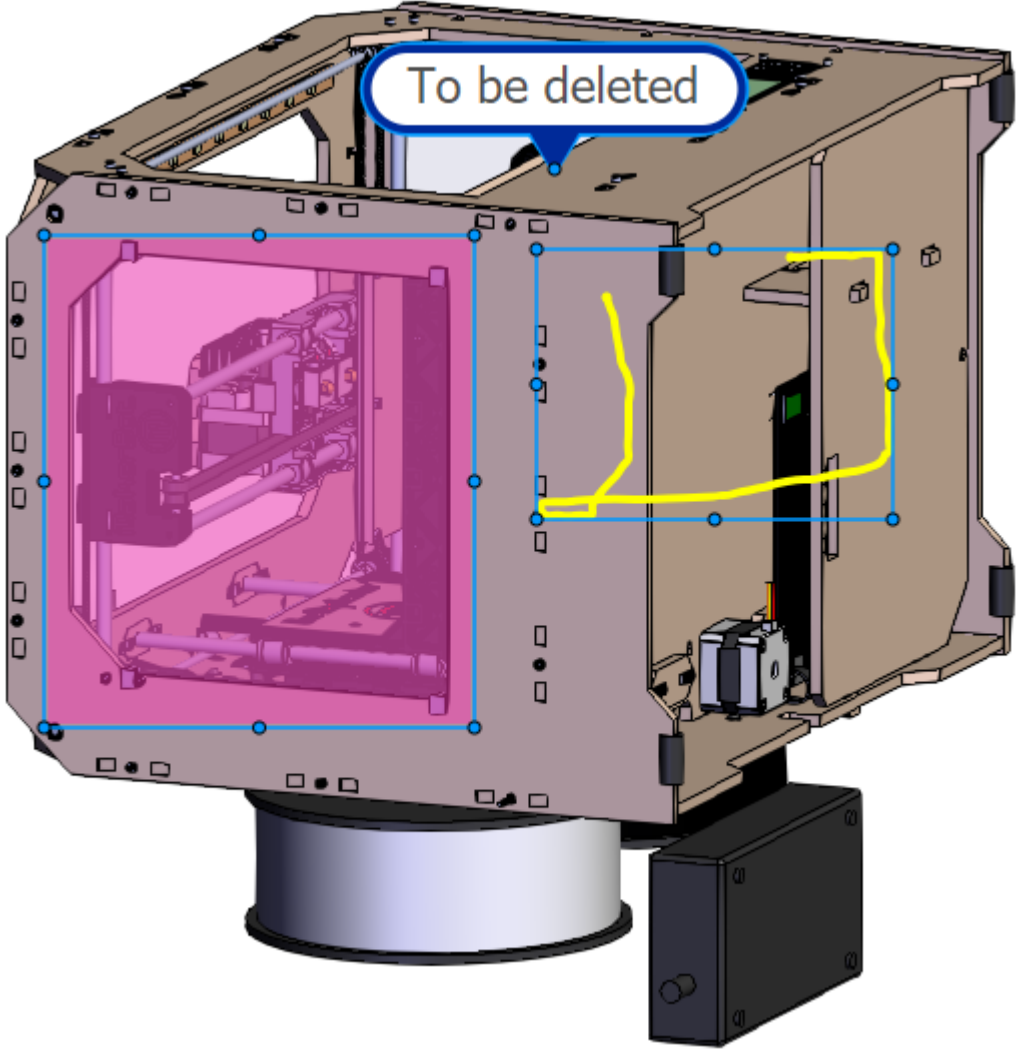
To select several markup items, a user should click **Select** on the **Basic Markup** or **Standard Markup** toolbar and then drag the cursor to specify the rectangular selecting area.





All the markup items within the selected area become selected.





## Deleting Markups on CAD Models

To delete one or more markup items:

Select the markup items. There are two ways to delete the Markup on CAD Models:

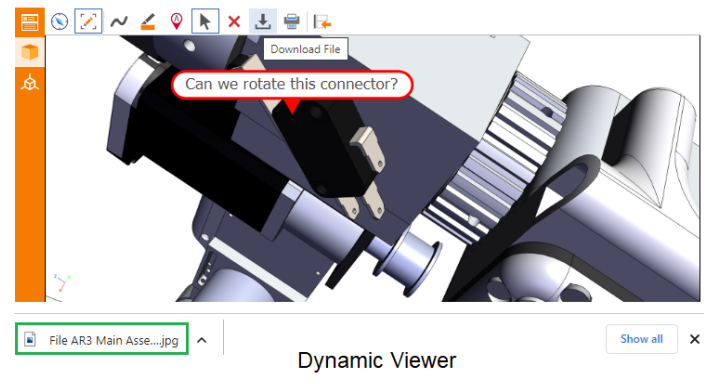
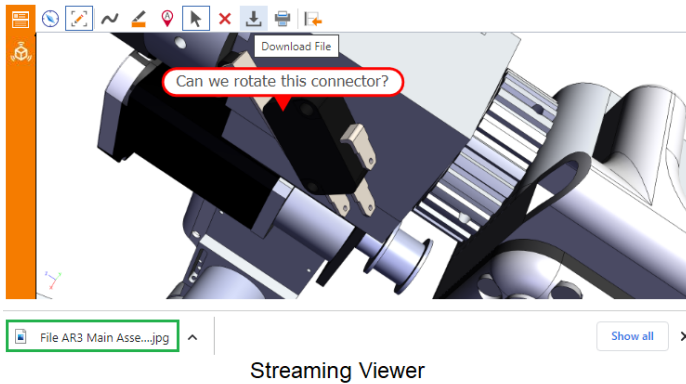
- Click Delete on a Markup toolbar.
- Press Delete on the keyboard.



## Saving and Sharing Markups

The options for saving and sharing a markup of a 3D CAD model are the following:

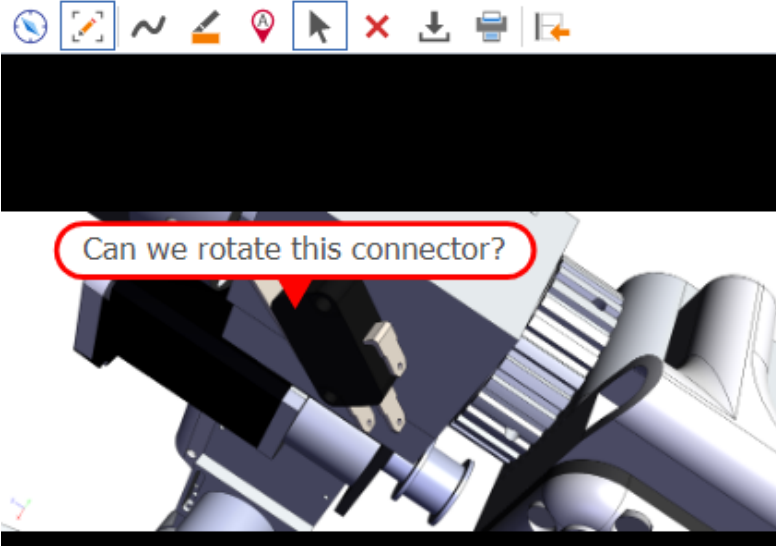
- Downloading a JPG image of the model view with annotations with the **Download File** button on the **StandardMarkup** toolbar.






- Printing a markup with annotations with the **Print** button on the **StandardMarkup** toolbar. Depending on a given browser and its specific configuration, various print choices, such as printing on a printer or saving to PDF, are available.






- Attaching a markup snapshot to a discussion message with the **Include snapshot** check box. See the Visual Collaboration in 3DV section.



Discussion  Search messages...  

Enter a comment...  
Use @ to mention people

Include snapshot  World  



## Monolithic Viewer

The Monolithic Viewer features are outlined within this section.

The Monolithic Viewer shows a static 3D CAD model of a component or assembly featuring all general Aras 3DV functionality for viewing and annotating 3D CAD models described in the General 3DV functionality section. Additionally, it can display product manufacturing information included in a given model as discussed in the Product Manufacturing Information subsection.

A 3D CAD model shown in the Monolithic Viewer is static because this Viewer shows technically a single viewable SCS file attached to a given CAD Document. For an assembly, such a file is created from the SCS files attached to all children of this CAD Document that were at the moment of loading into the Viewer. The Monolithic Viewer benefits are lightweight rendering and the ability to display product manufacturing information and configurations in some cases. The main Monolithic Viewer limitations are the lack of support for resolutions besides as-saved. The Dynamic and the Streaming Viewer resolves this disadvantage by supporting various structure resolution, such as the latest, and has several other enhancements, such as custom rendering, Saved Views, and the Digital Mockup support.

The Monolithic Viewer fully reuses the 3D scene canvas, 3DV toolbars, and context menus described in the Aras 3DV UI section. It also features the simple Model Browser for browsing assembly parts and model views as discussed in the Model Browser in Monolithic Viewer subsection.

The Monolithic Viewer does not support multi-selection: only one subassembly, part, or surface can be selected during viewing.

## Product Manufacturing Information

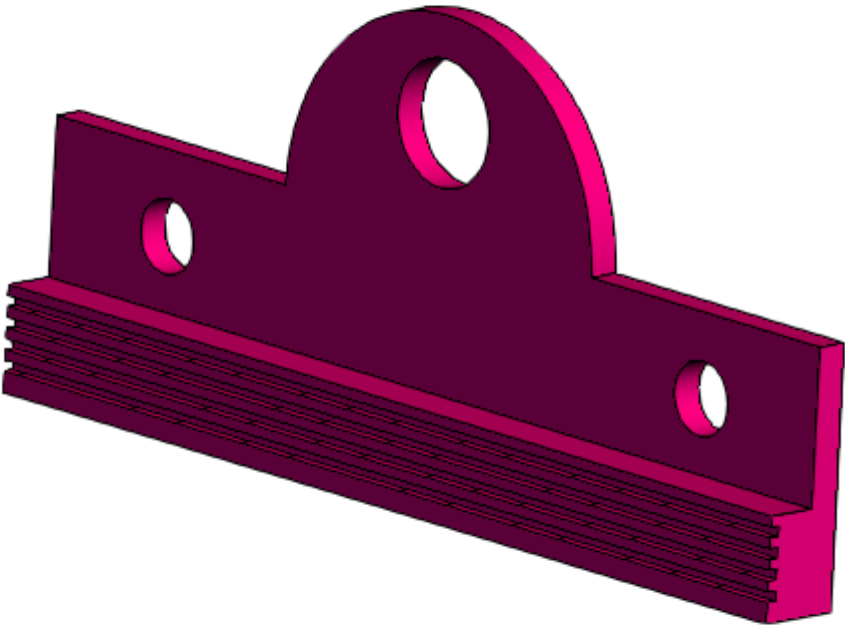
The **Product Manufacturing Information** button on the **Standard Viewing** toolbar provides access to the product manufacturing information (PMI) included in a 3D CAD model.



PMI is non-geometric attributes embedded in a 3D CAD file that convey various non-geometric data necessary for manufacturing parts and assemblies, like geometric dimensions and tolerances, 3D annotations (text), surface finishes, material specifications, and so on. It combines a model-based definition with a 3D model eliminating 2D drawings or digital documents to provide engineering or manufacturing data.

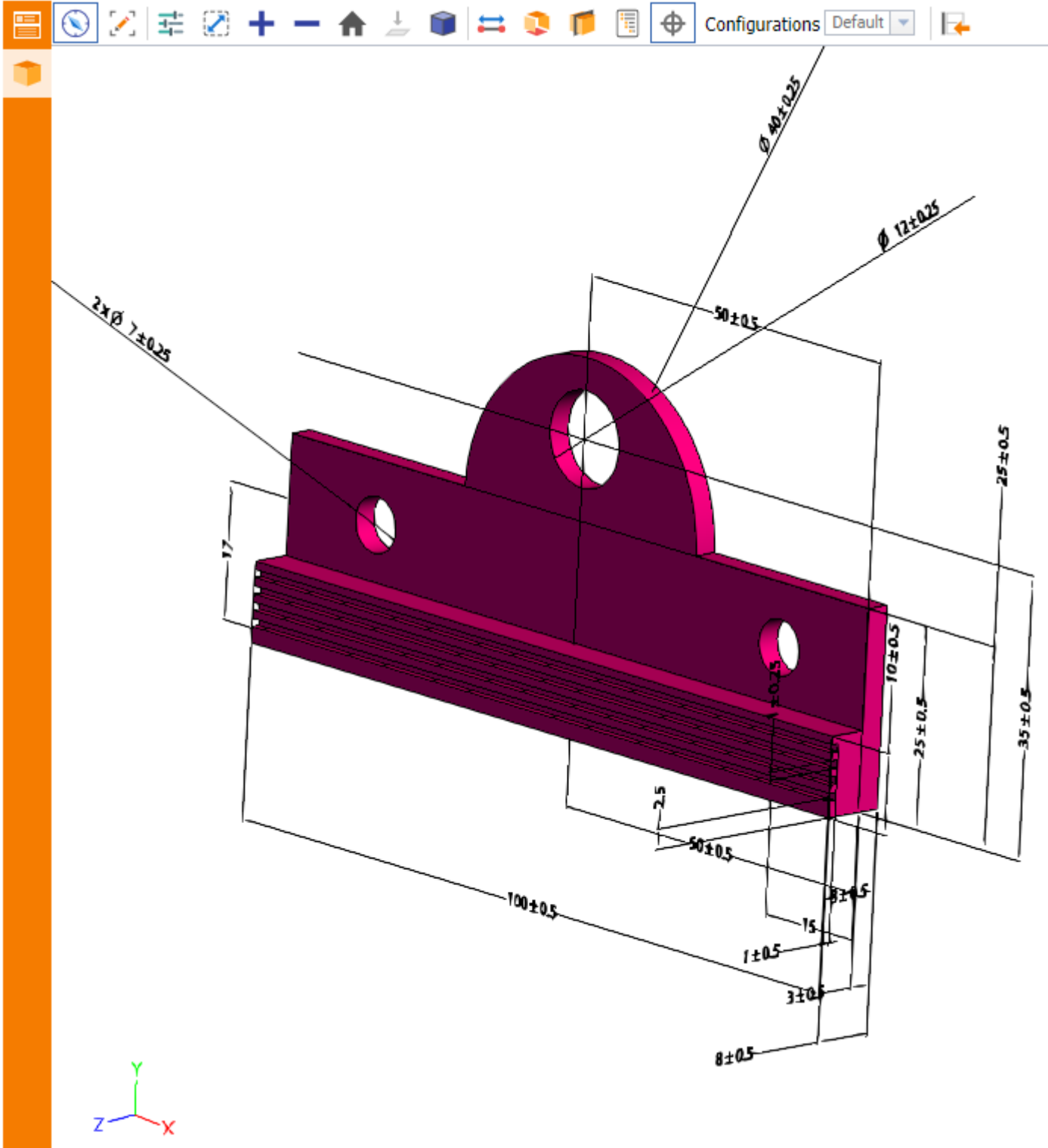
PMI is typically stored at the assembly level within native CAD files and is included by default in the process of converting the native CAD files into viewable SCS files. Consequently, the Monolithic Viewer can show a 3D model and PMI that are both sourced from an SCS file.

By default, a 3D CAD model is shown without its PMI.



Clicking the **Product Manufacturing Information** button shows or hides the model PMI.





When shown, graphic PMI may be positioned such that it is hard to read or interferes with some other rendered 3D component geometry. An end user should rotate a model to view and read a given PMI piece conveniently.

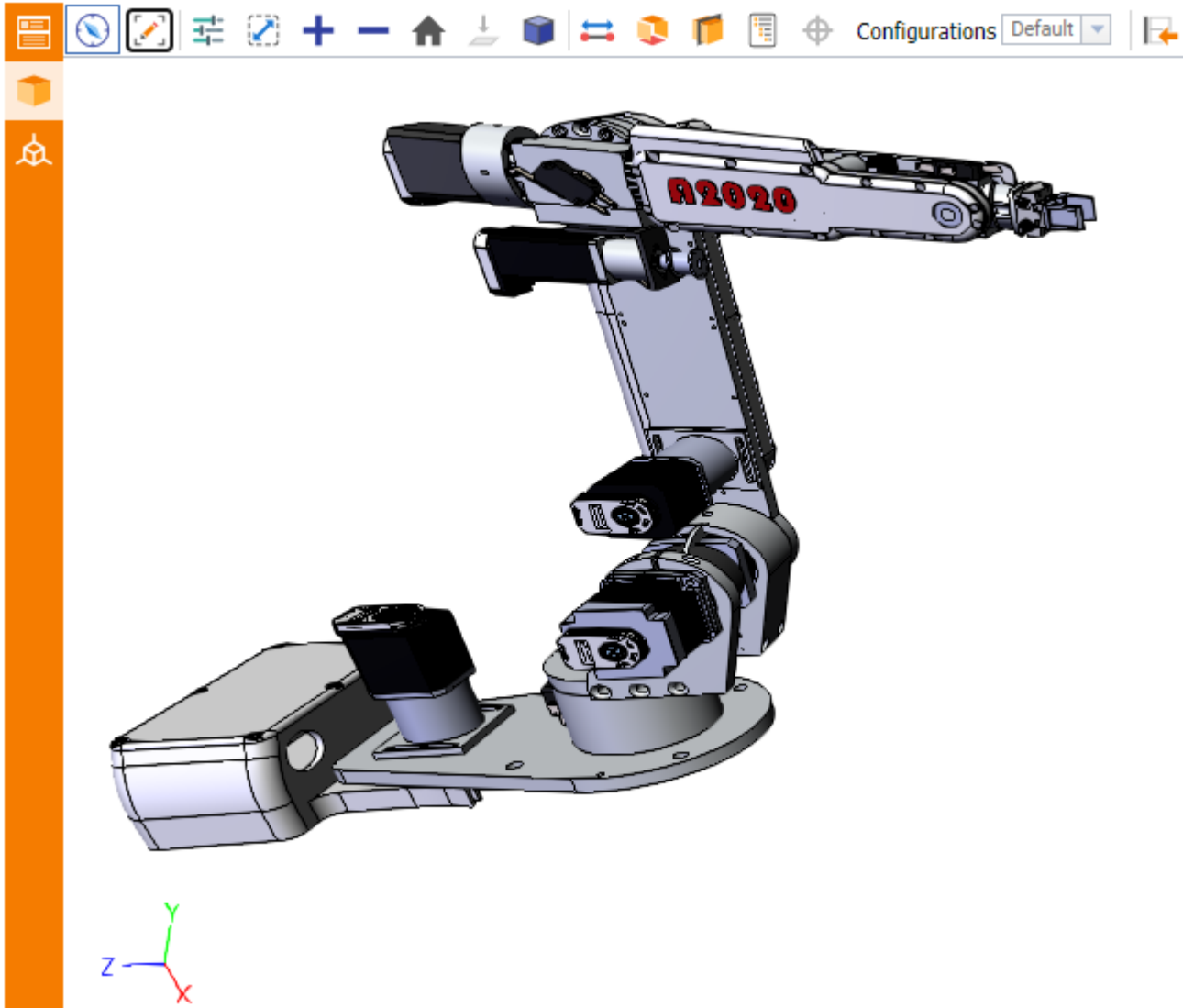
The shown PMI reflects the current CAD View. To view PMI attached to a different CAD View, select it on the **Views** tab.

## Model Browser in Monolithic Viewer

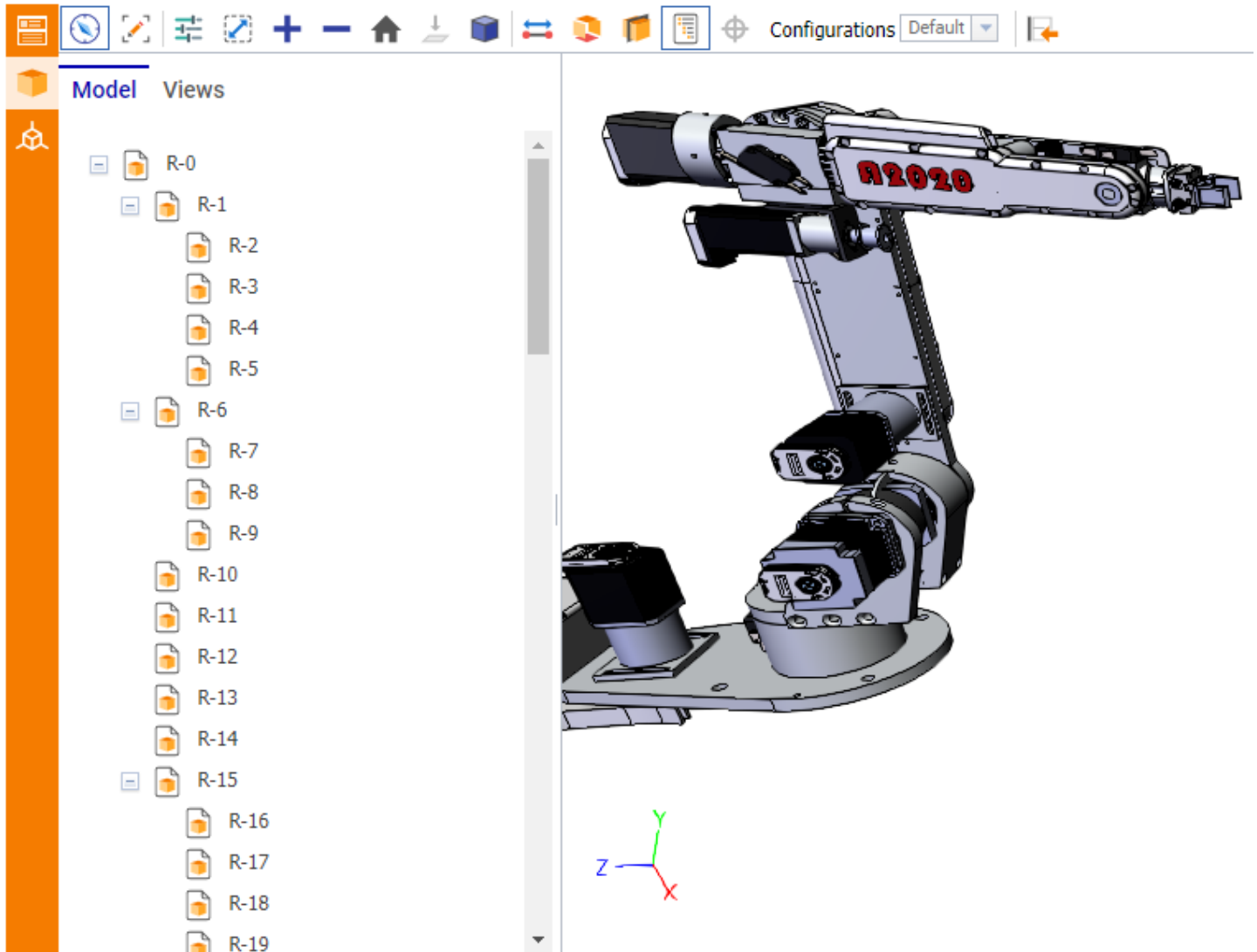
The **Model Browser** button on the **Standard Viewing** toolbar provides access to the **Model Browser** section.



By default, the **Model Browser** section is hidden at the first opening of the Monolithic Viewer for a given CAD Document in the current logging session.



Clicking the **Model Browser** button shows or hides the **Model Browser** section right after the left sidebar.



The **Model Browser** section has two tabs:

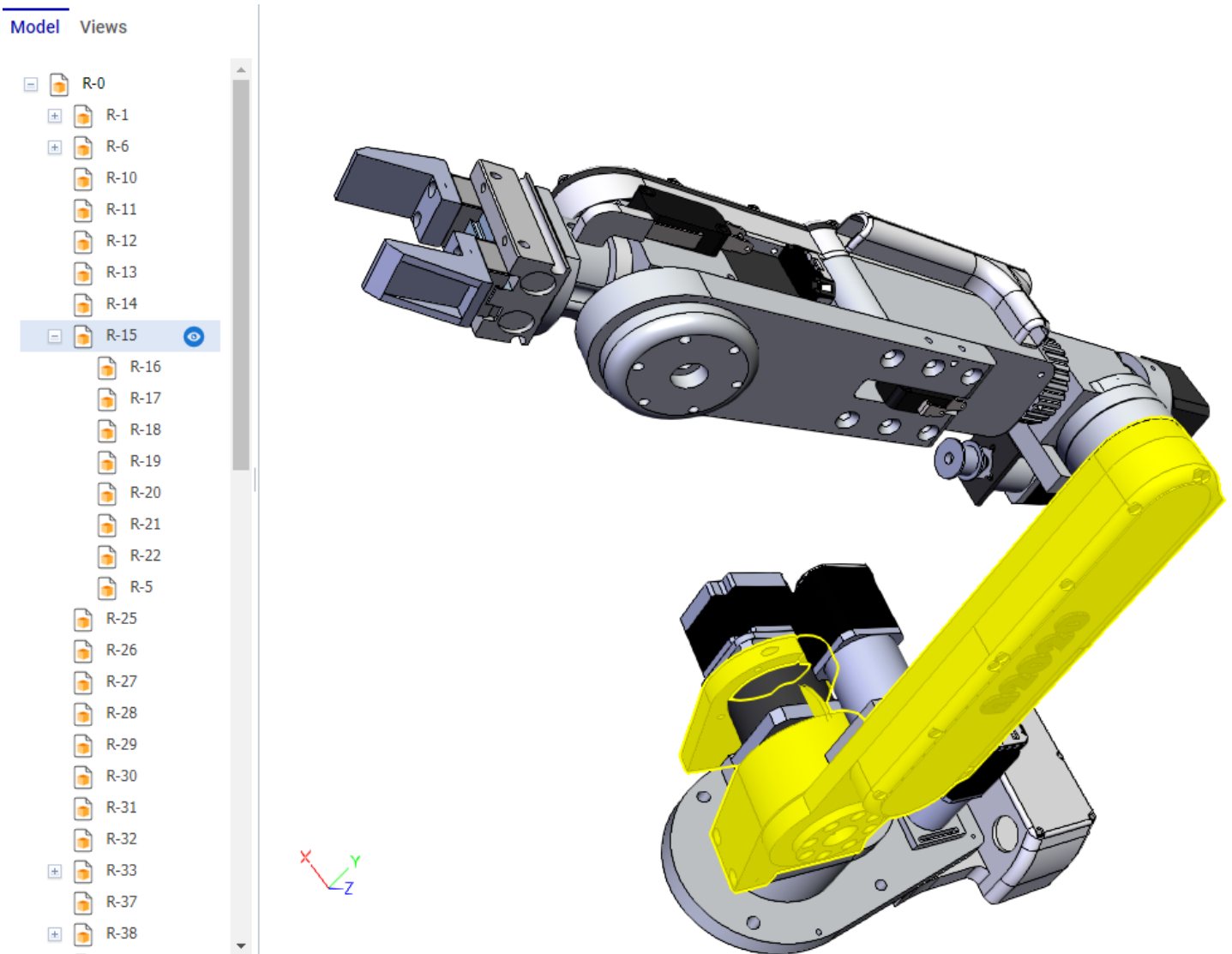
- **Model:** a multi-level tree of CAD Documents included in a CAD assembly representing a given 3D CAD model with capabilities to:
  - Navigate through the subassemblies and parts included in the given assembly.
  - Quickly select a necessary subassembly or part.
- **Views:** a list of model views to switch quickly between viewing positions available for a given 3D CAD model.

### Browsing CAD Model Assemblies and Parts with Monolithic Viewer

The **Model** tab of the **Model Browser** section provides end users with quick navigation through an assembly structure shown as a tree of CAD Documents.

### Selecting Parts in Model Browser

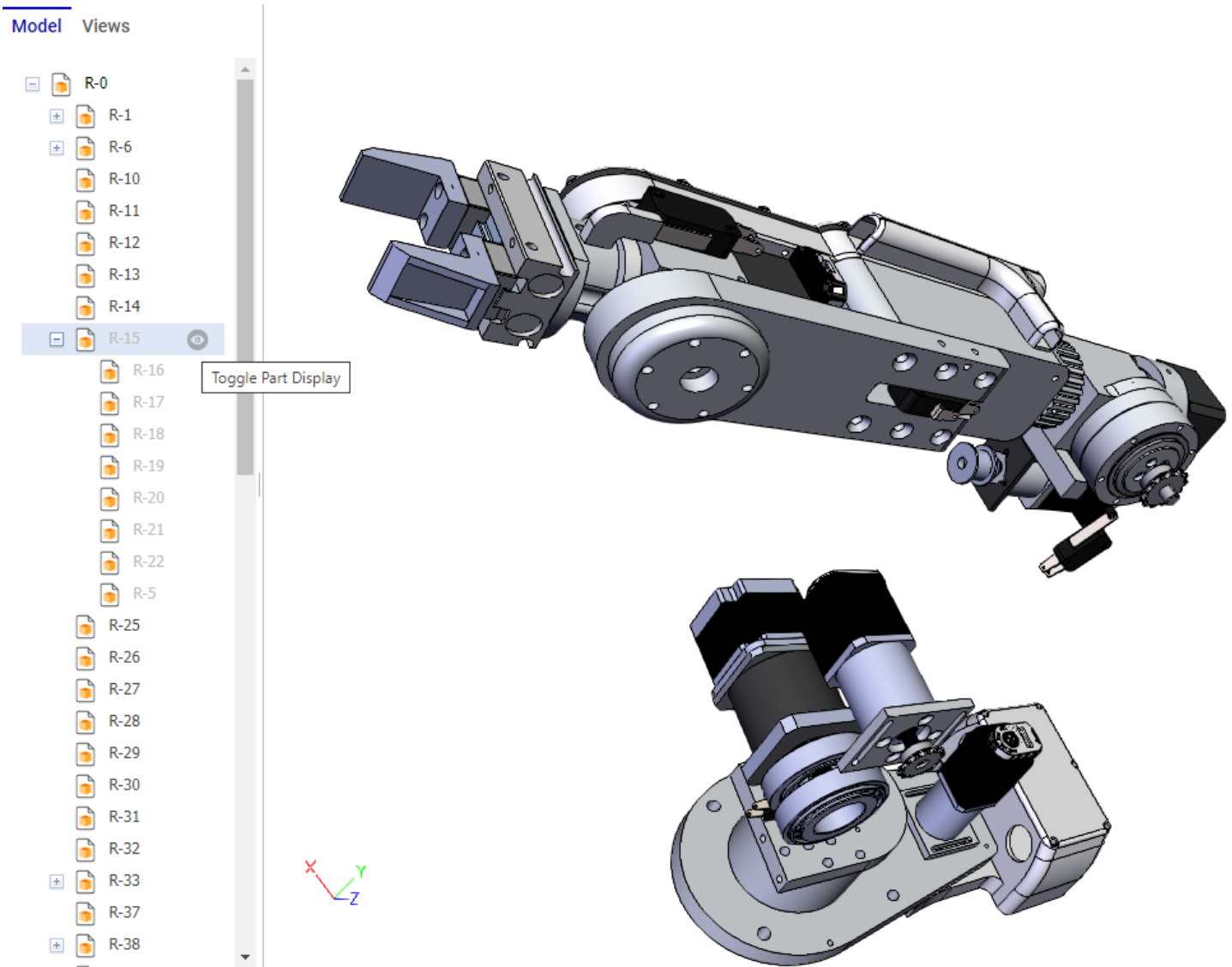
Part selection is synchronized between the CAD Document tree on the **Model** tab of **Model Browser** and a given 3D CAD model. Clicking a part or surface on the model selects this part on the model and its CAD Document in the tree. And vice versa, clicking a CAD Document in the tree selects its part or assembly on the model as well.



### Hiding and Unhiding Parts with Model Browser

Each CAD Document in the **Model** tab of **Model Browser** has the **Toggle Part Display** button to hide or show its represented part or assembly on the model. CAD Documents of hidden parts and assemblies are light grey in the tree.

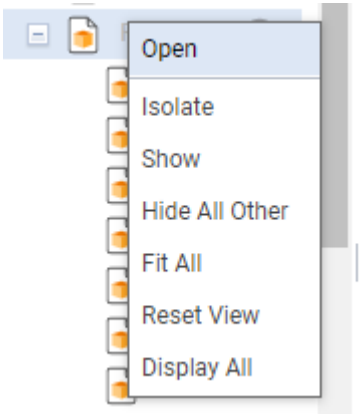




### Part Context Menu in Model Tree

Right-clicking a CAD Document in the **Model** tab of **Model Browser** displays the **Part** context menu for this CAD Document and its represented part or assembly.

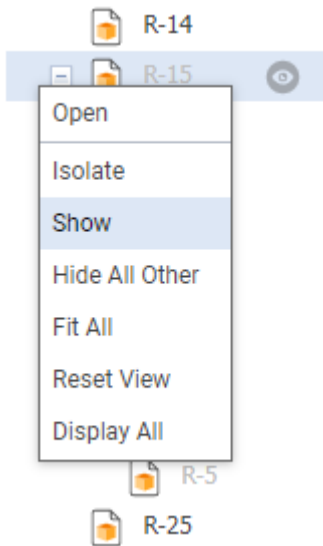




The **Model** tree fully reuses the **Part** context menu, its commands, and their behavior as discussed in the Part Context Menu section. The only exception is the **Hide** command, which is:

- Shown for a CAD Document whose part or assembly is visible on a 3D CAD model.
- Replaced by the **Show** command for a CAD Document whose part or assembly is hidden on a 3D CAD model.

Clicking the **Show** command makes a given hidden part or assembly visible on a 3D CAD model.



### Browsing CAD Model Views with Monolithic Viewer

The **Views** tab of the **Model Browser** section provides end users with options of viewing positions available for a given 3D CAD model.



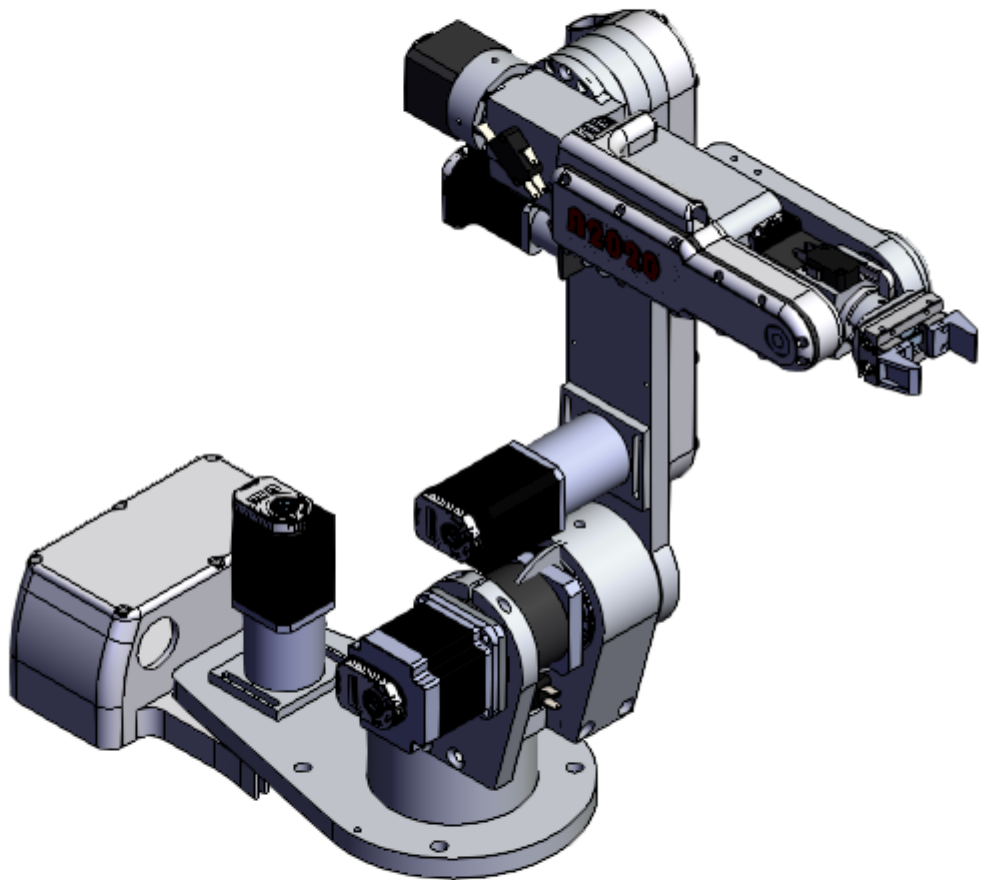
Model Views

Standard Views

- Iso View
- Front View
- Right View
- Left View
- Top View
- Back View
- Bottom View

CAD Views

- [\*Normal To]
- [\*Front]
- [\*Back]
- [\*Left]
- [\*Right]
- [\*Top]
- [\*Bottom]
- [\*Isometric]
- [\*Trimetric]
- [\*Dimetric]
- \*Back
- \*Right
- \*Bottom
- Home



The **Views** tab has two view groups:

1. **Standard Views**: viewing positions of standard 3D sides available for all 3D CAD models:



- Iso View
- Front View
- Right View
- Left View
- Front View
- Top View
- Back View

2. **CAD Views:** viewing positions of specific sides and orientations embedded in a given 3D CAD model with a CAD editor.

Clicking a given **viewing** position rotates a given 3D CAD model to a corresponding side and orientation.



## Dynamic and Streaming Viewer

### Dynamic Viewer

The Dynamic Viewer is based on and requires the Monolithic Viewer.

A 3D CAD model shown in the Dynamic Viewer is dynamic because this Viewer shows an assembly dynamically generated on the fly from multiple viewable SCS files attached to the Items, such as CAD Documents, which were the children of the context Item, such as a CAD Document, representing an assembly at the moment of loading into the Viewer. This is also referred to as “shattered” viewing.

The Dynamic Viewer limitations are as follows:

- Not available for single components without structure.
- No support for product manufacturing information included in a 3D CAD model.
- No support for configurations.

The Monolithic Viewer resolves these limitations.

### Streaming Viewer

A 3D CAD model shown in the Streaming Viewer is also dynamic as the viewer shows an assembly dynamically generated from multiple viewable SCZ files.

The Streaming Viewer enables user to stream large datasets from a server to a client in a single request, that, in some cases, provides an improved response time than multiple individual requests.

The Streaming Viewer limitations are as follows:

- No support for product manufacturing information included in a 3D CAD model.
- No support for configurations.
- No support for BLOB Storages on cloud environments.
- Only one Streaming Viewer can be installed on one machine at a time.

#### Important

The Dynamic and Streaming Viewers are not compatible with each other. Only one of the two viewers can be installed.



If the Streaming 3D Viewer is installed in an environment where the Monolithic or Dynamic 3D Viewers have been previously installed and used, all existing native files of existing CAD Documents need to be re-converted. There is no automated means to perform this reconversion.

### Important

The Streaming Viewer currently cannot be deployed in a cloud environment.

The HOOPS Server must be deployed with networked file access to a single vault containing view files for rendering.

Users can do the following with the Dynamic and Streaming Viewers:

- Manually move, rotate, and mark up geometry from the Viewer.
- Add and remove additional parts and assemblies from the 3D scene using Digital Mockup.
- Configure what is retrieved from the CAD structure and how it is shown in the Viewer.

There are out-of-the-box features specific to the Dynamic and Streaming Viewers—additional Aras 3DV functionality for displaying, annotating, and collaborating on a 3D CAD model.

It shows a dynamic 3D CAD model of an assembly featuring all general Aras 3DV functionality for viewing and annotating 3D CAD models described in the *General 3DV functionality* section. In addition to the general functionality, it provides its own features described in this section.

The Dynamic and Streaming Viewers fully reuse the 3D scene canvas and 3DV toolbars outlined in the 3 Aras 3DV UI section. It also extends the general 3DV context menus as outlined in the *Dynamic Viewer* and *Streaming Viewer* Context Menu subsection.

The extended 3DV context menus provide functionality for:

- Adding markup lines onto a 3D CAD model as outlined in the Markup Lines in Dynamic Viewer and Streaming Viewer subsection.
- Transforming geometries of parts and subassemblies as outlined in the *Manual Geometry Transformation* subsection.

The outlined viewer features the Tree Grid View (TGV) Model Browser for browsing assembly parts and standard model views as outlined in the Model Browser in Dynamic Viewer and Streaming Viewer subsection. Users can also save and use custom model views.



It also supports multi-selection: one or more subassemblies, parts, and surfaces can be selected during viewing as discussed in the *Multi-Selection* subsection.

Administrators of the Aras Innovator platform can customize the Dynamic and Streaming Viewers. For more information on the customization see, *Aras 3D Visualization 35 - Administrator Guide*.



## Dynamic Viewer and Streaming Viewer Context Menus

The Dynamic Viewer and the Streaming Viewers fully reuses the general **3D Scene** and **Part** context menus outlined in the Aras 3DV Context Menu section. It also extends these context menus with commands discussed in this section.

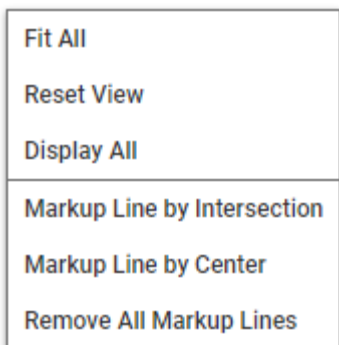
### 3D Scene Context Menu in Dynamic Viewer and Streaming Viewer

The **3D Scene** context menu in the Dynamic and Streaming Viewers fully reuses the general **3D Scene** 3DV context menu, its commands, and behavior as outlined in the 3D Scene Context Menu section.



In addition to the general commands, the 3D Scene Dynamic and Streaming Viewer context menu includes the following ones:

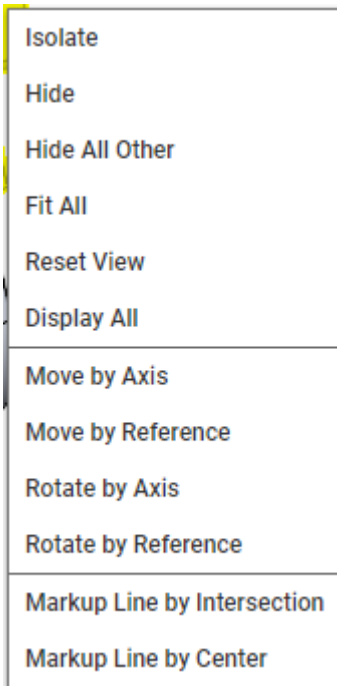
- **Markup Line by Intersection:** to add a markup line between two arbitrary points on a 3D CAD model; see the *Adding Markup Lines by Intersection* section.
- **Markup Line by Center:** to add a markup line between the centers of two arbitrary edges on a 3D CAD model; see the *Adding Markup Lines by Center* section.
- **Remove All Markup Lines:** to clear a 3D CAD model from all markup lines; see the *Removing Markup Lines* section. This button is dynamic: it is displayed only when one or more markup lines exist on the model. If there are no markup lines, the button is absent on the menu.



## Part Context Menu in Dynamic and Streaming Viewers

The **Part** context menu in the Dynamic and Streaming Viewers fully reuse commands and behavior from the following context menus:

- General **Part** 3DV context menu as outlined in the *Part Context Menu* section.
- Custom **3D Scene** Dynamic and Streaming Viewer context menu as outlined in the *3D Scene Context Menu in Dynamic Viewer and Streaming Viewer* section.



In addition to the reused commands, the Part Dynamic and Streaming Viewers context menu includes the following ones:

- **Move by Axis:** to move a part or subassembly selected on a parent assembly 3D CAD model along one or more axes in the coordinate system of this part or subassembly; see the *Moving Parts by Axis* section.
- **Move by Reference:** to move a part or subassembly selected on a parent assembly 3D CAD model along a surface or edge of another part or subassembly; see the *Moving Parts by Reference* section.
- **Rotate by Axis:** to rotate a part or subassembly selected on a parent assembly 3D CAD model along one or more axes in the coordinate system of this part or subassembly; see the *Rotating Parts by Axis* section.
- **Rotate by Reference:** to rotate a part or subassembly selected on a parent assembly 3D CAD model along a surface or edge of another part or subassembly; see the *Rotating Parts by Reference* section.

## Digital Mockup

In Aras 3DV, a digital mockup is an ad-hoc arrangement of 3D component geometry on the 3D scene for analysis, review, or other purposes. End users can visualize collections of 3D assemblies, subassemblies, and parts in a manner that may be different from how these objects were defined within a CAD editor.

Using the Dynamic or Streaming Viewers digital mockup features, the end users can:

- Place additional assemblies, subassemblies, and parts onto a single 3D scene as outlined in the *Adding Additional Models to 3D Scene* section.
- Manipulate the position, orientation (by 3D rotation), and display of assemblies, subassemblies, and parts on a single 3D scene as outlined in the *Manual Geometry Transformation* section.
- Annotate assemblies, subassemblies, and parts on a single 3D scene with markup lines as outlined in the *Markup Lines in Dynamic and Streaming Viewers* section.
- Store an ad-hoc 3D scene view of a digital mockup for future use as outlined in the *Saved Views* section.
- Share and restore a digital mockup from a snapshot in the discussion panel as discussed in the *Visual Collaboration* section.



## Markup Lines in Dynamic and Streaming Viewers

In the Dynamic and Streaming Viewers, a markup line is a red dashed line that connects two arbitrary surfaces or edges for illustrative purposes, like marking spots that should be fitted.

The Dynamic and Streaming Viewers has two markup line types:

- Between two arbitrary points on any surfaces or edges.
- Between two arbitrary edges. In this case, a line is automatically placed between the centers of given edges, which can have different shapes.

While viewing one or more 3D CAD models in the Dynamic Viewer or in the Streaming Viewer, an end user can annotate these models with one or more markup lines of any type as outlined in this section. The end user can also combine markup lines with manual geometry transformation of the models that is outlined in the *Manual Geometry Transformation* section.

The Dynamic and Streaming Viewers do not embed markup lines into a 3D CAD model native and viewable files.

A 3D CAD model view with markup lines can be saved and shared.

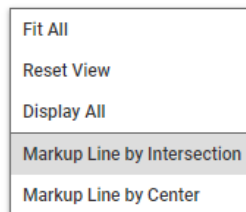
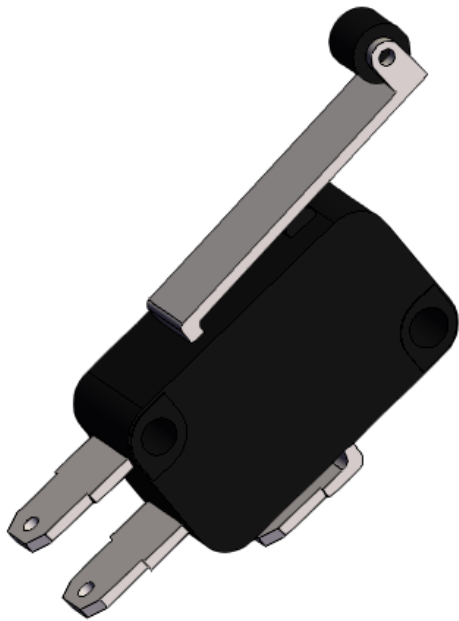
To quit adding a markup line, press Escape on the keyboard.



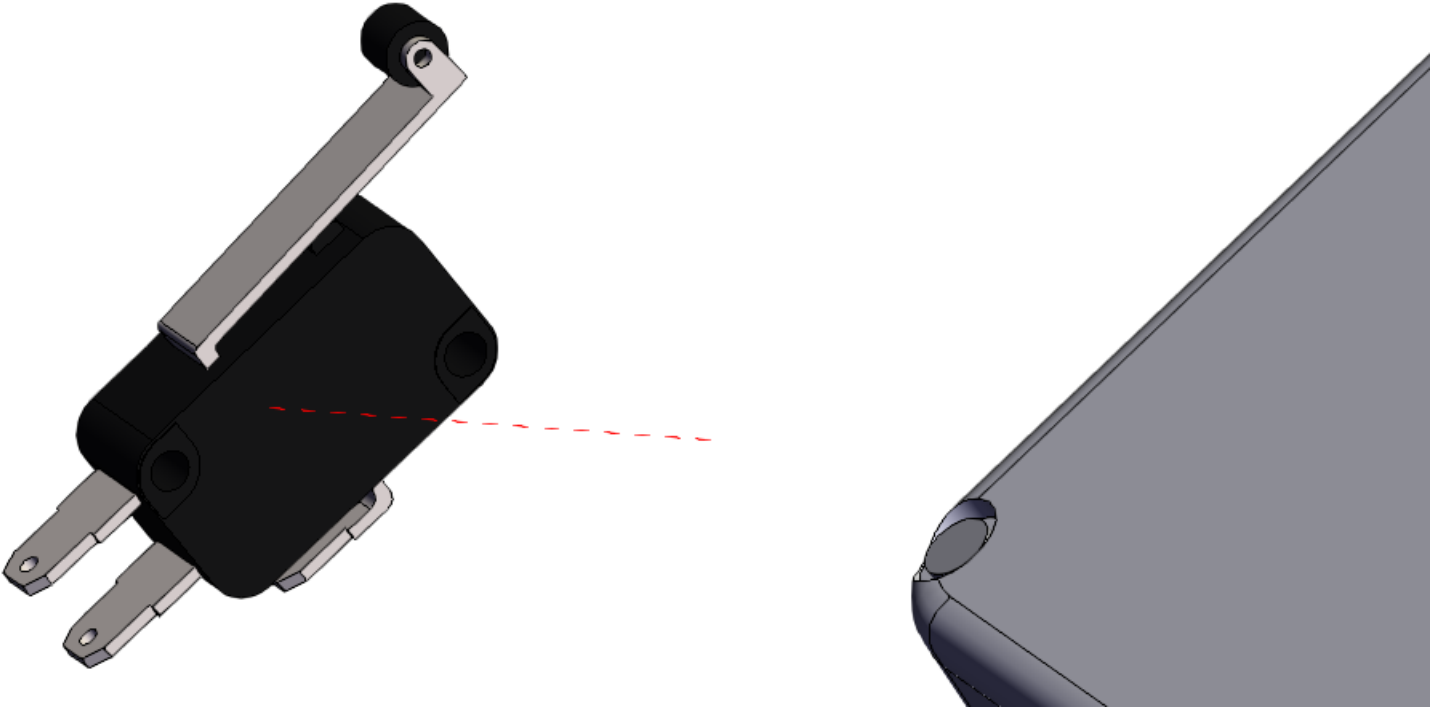
### Adding Markup Lines by Intersection

The following process outline the process to draw a markup line between two arbitrary points on one or more 3D CAD models:

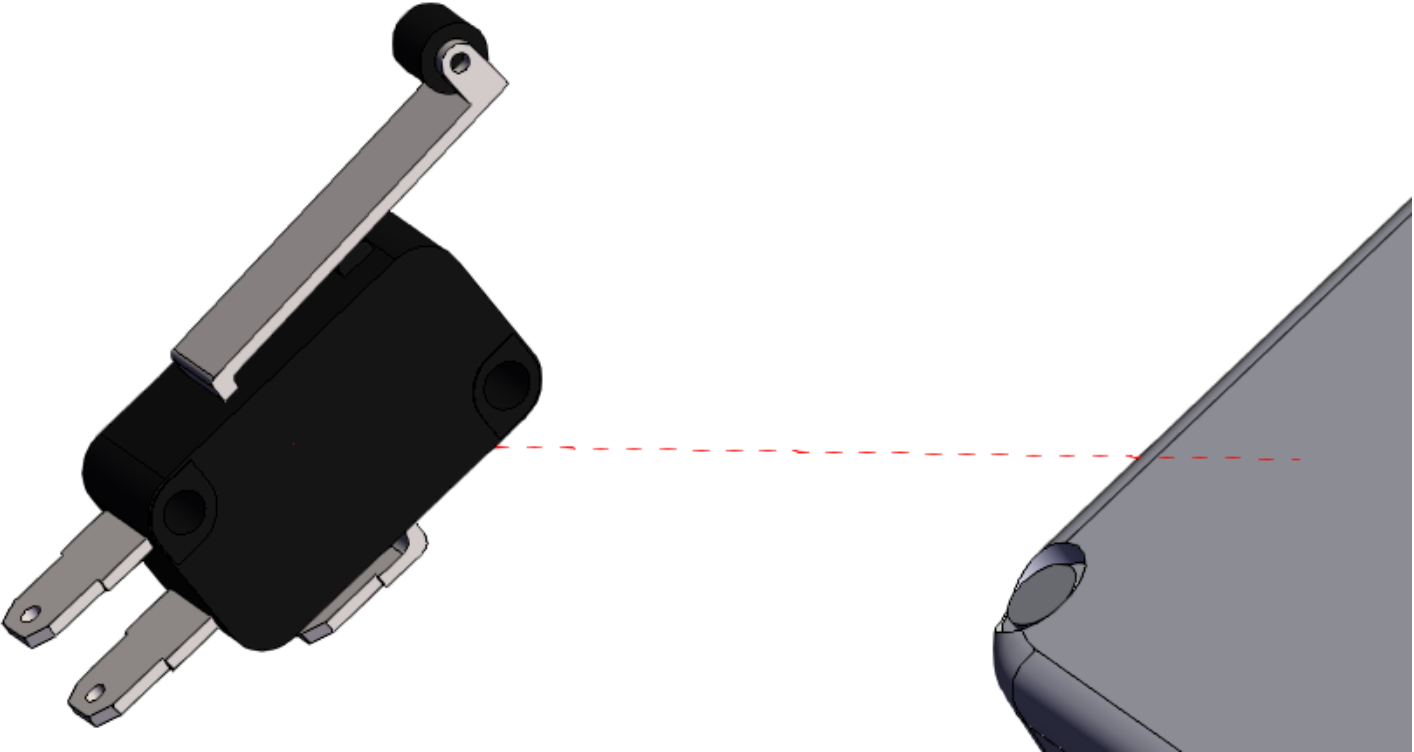
1. Right-click anywhere on the 3D scene.
2. Click **Markup Line by Intersection**.



3. Click on the first point.



4. Click on the second point. The markup line is created.

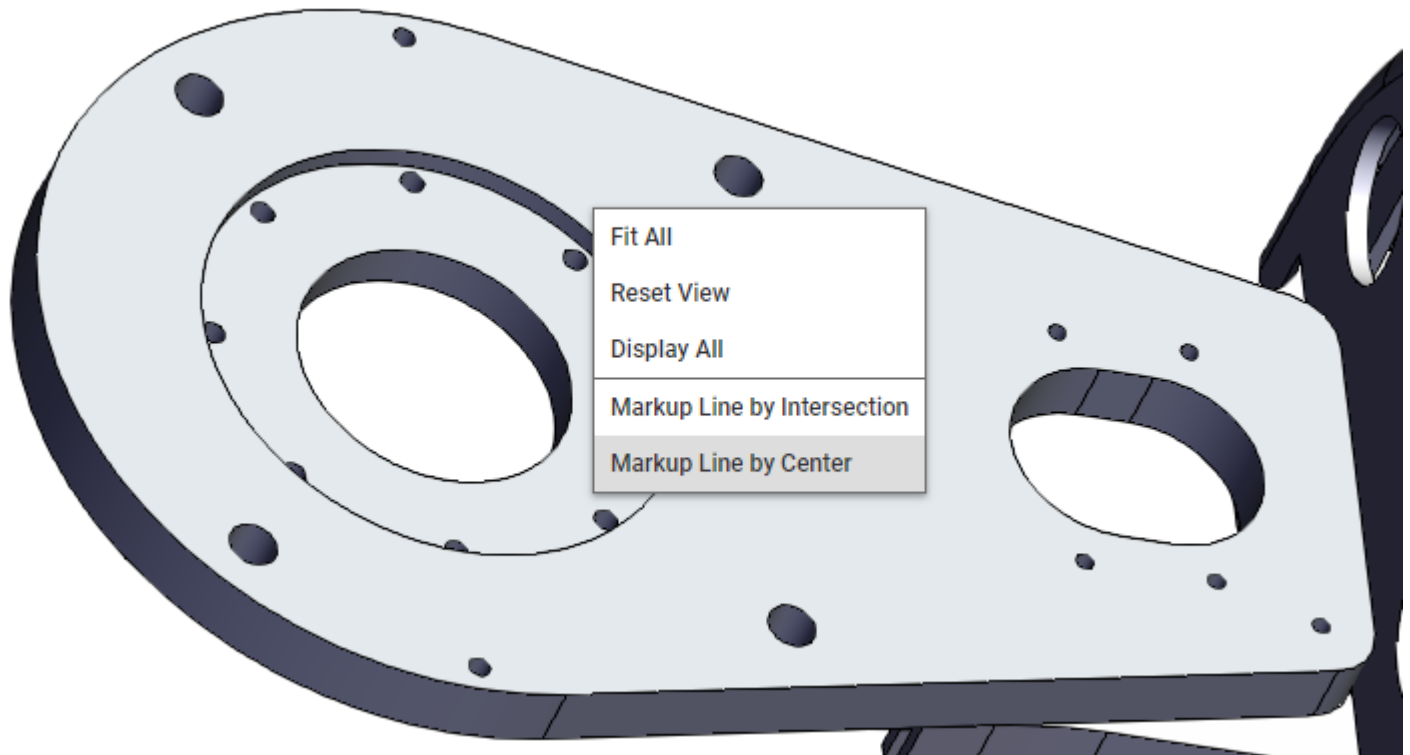




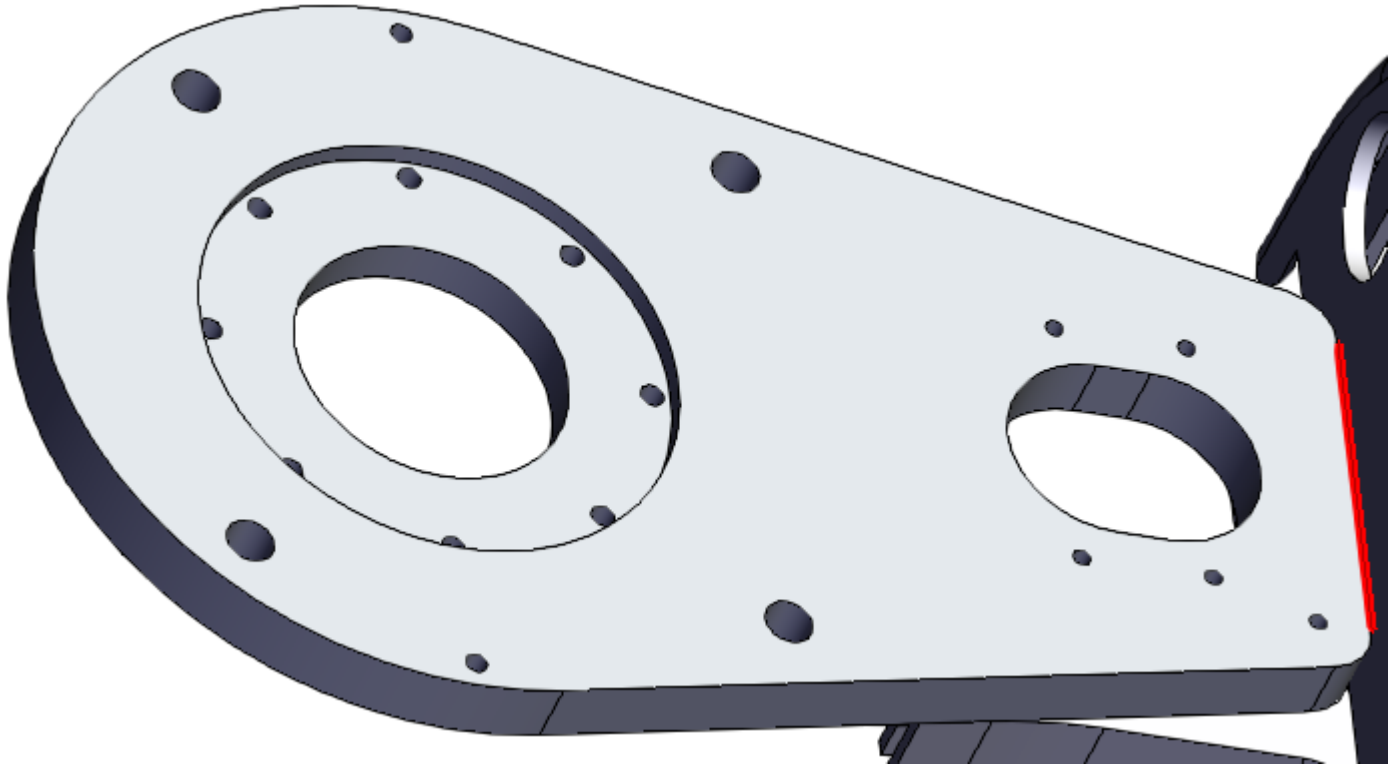
### Adding Markup Lines by Center

The following steps outline the process to draw a markup line between the centers of two arbitrary edges on one or more 3D CAD models:

1. Right-click anywhere on the 3D scene.
2. Click **Markup Line by Center**.

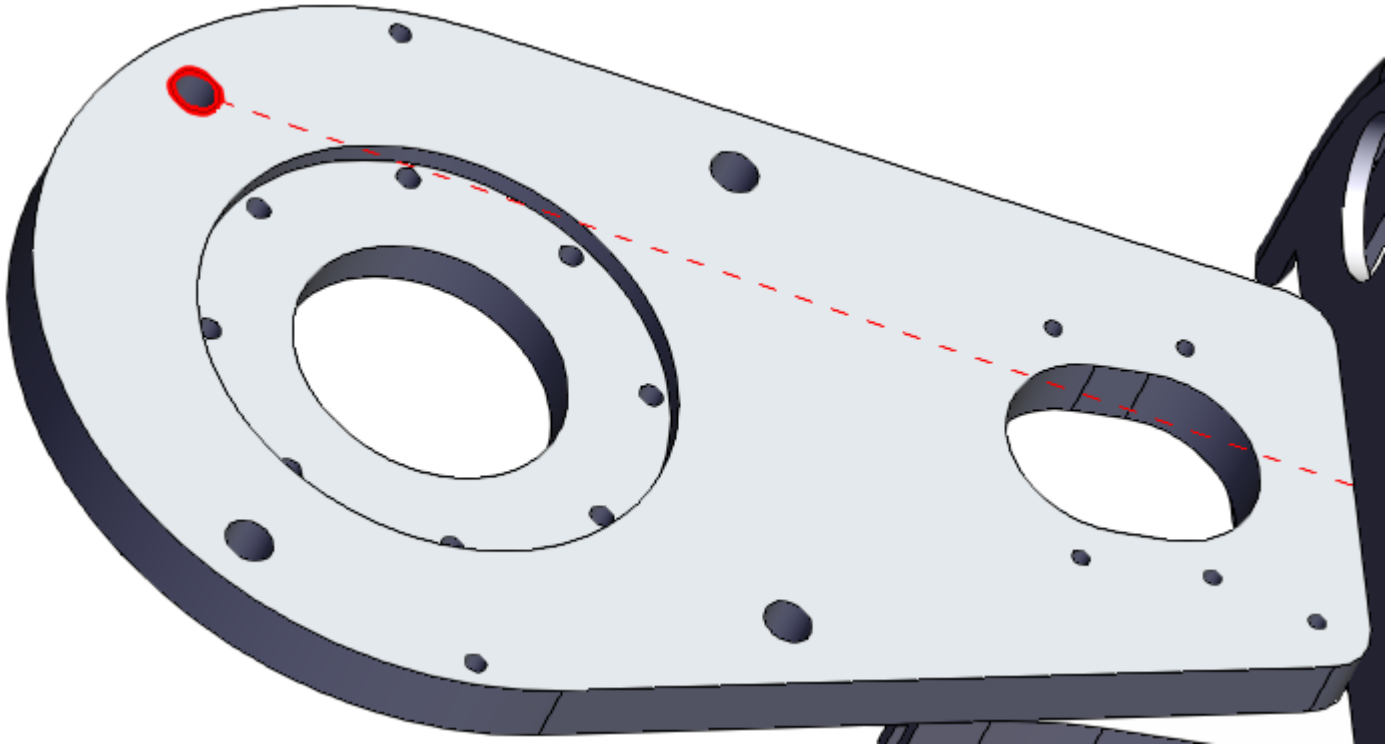


3. Click on the first edge.

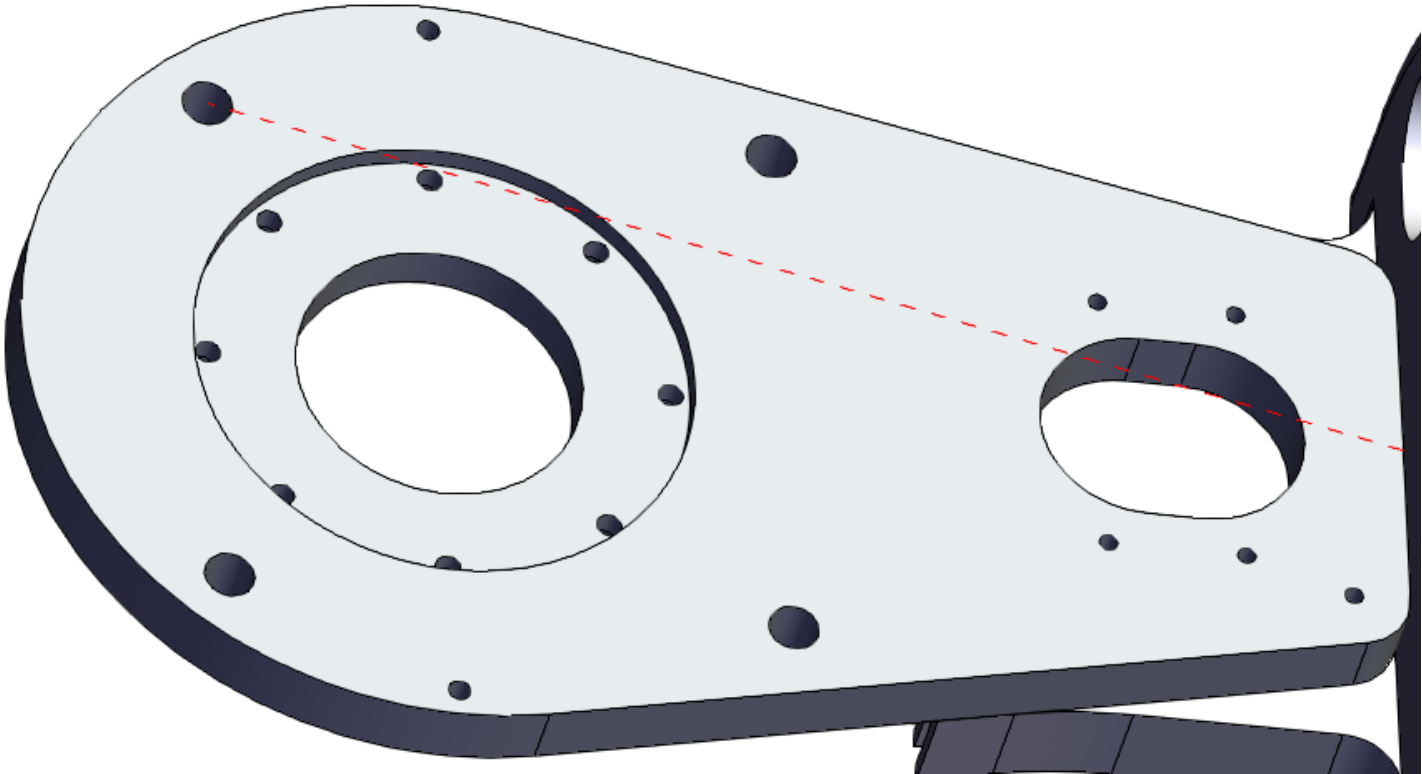


**Important**

Edges available for selection are highlighted with red while hovering over them.

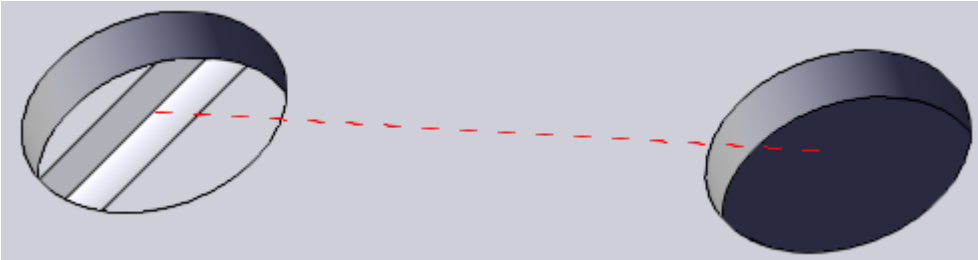


4. Click on the second edge. The markup line between the centers of the two edges is created.



## Removing Markup Lines

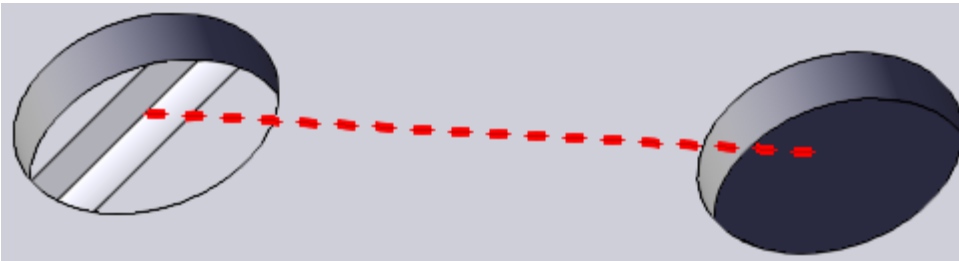
An end user can remove a single markup line or clear given 3D CAD models from all lines with a single command.



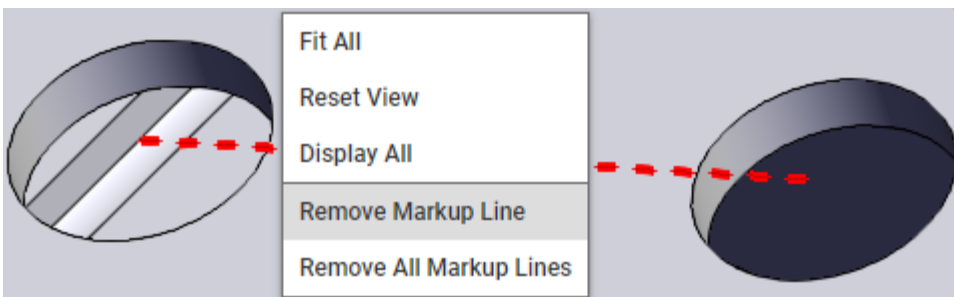
## Removing Single Markup Line

The following steps outline the process to remove an existing markup line on one or more 3D CAD models:

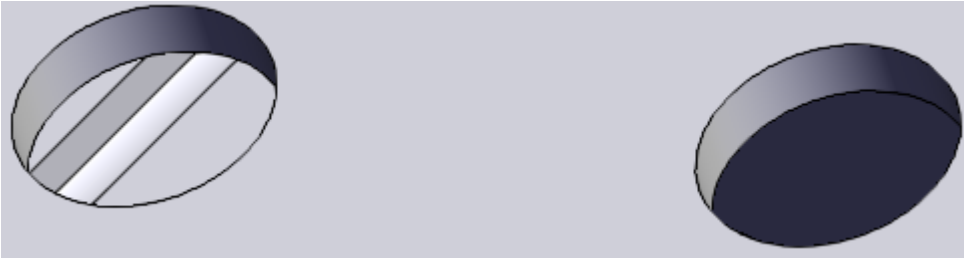
1. Click the markup line to be removed. The line becomes bold.



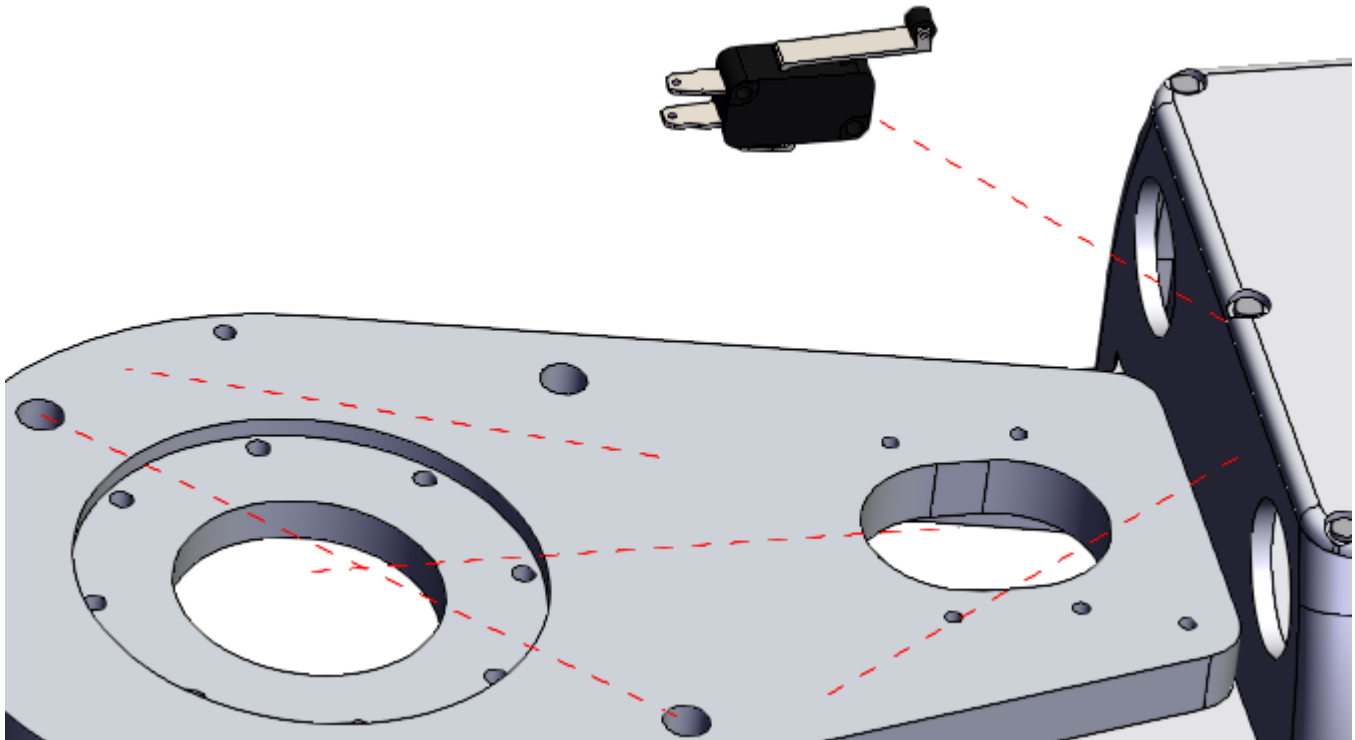
2. Right-click and then click **Remove Markup Line**.



The markup line is removed from the 3D scene and cannot be restored.



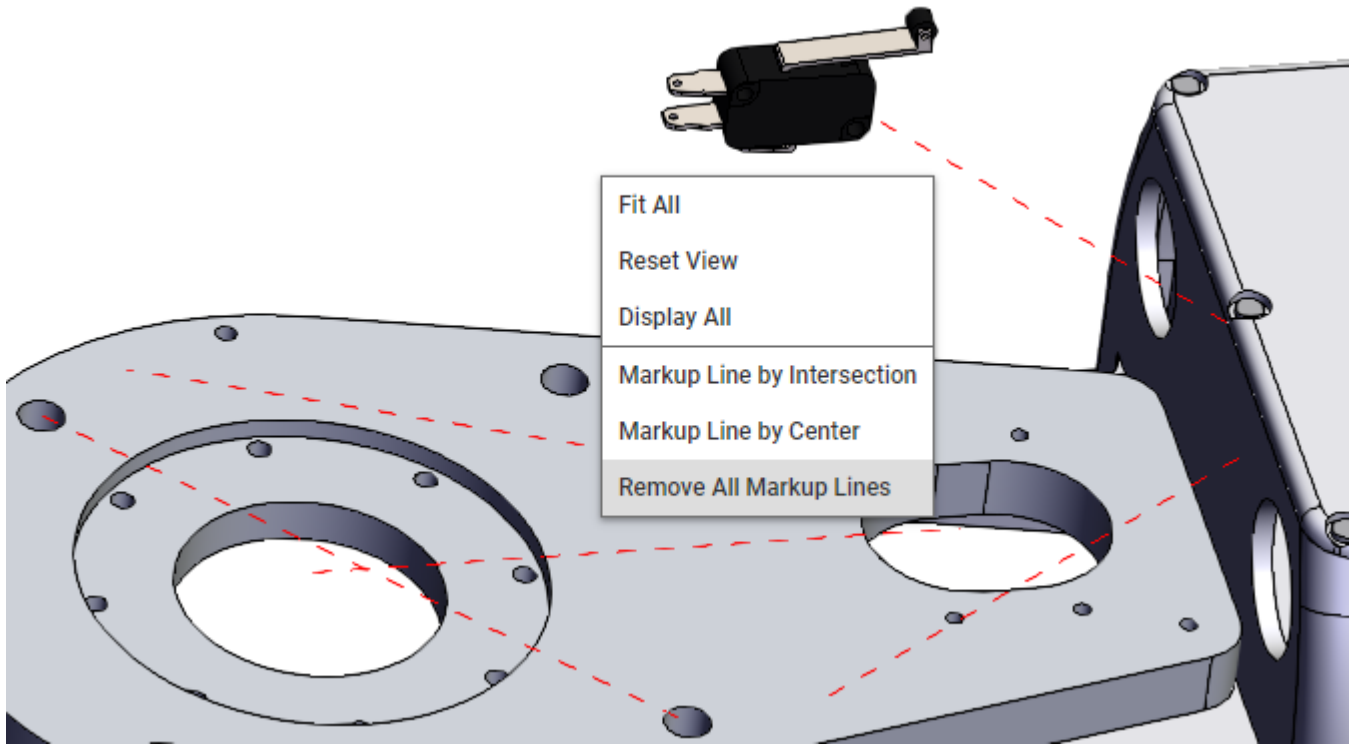
### Clearing 3D Scene from All Markup Lines



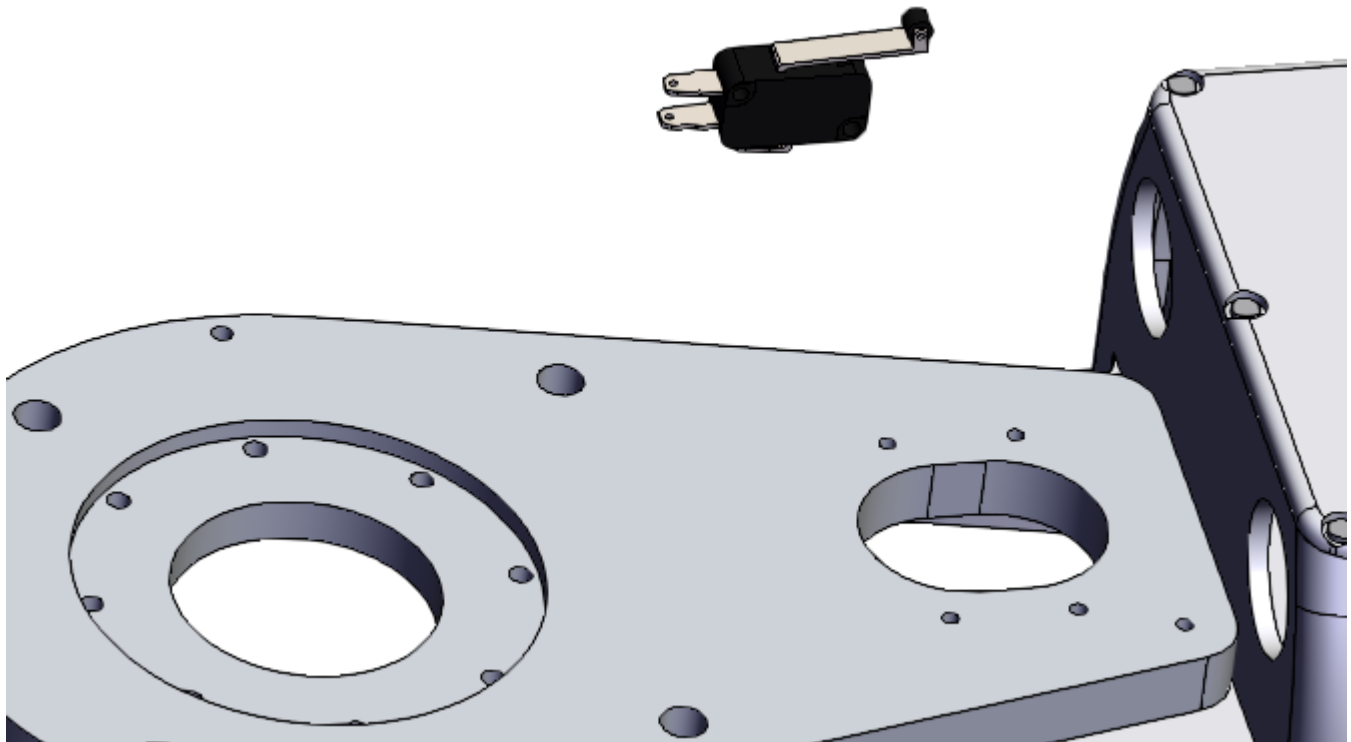
The following steps outline the process to clear 3D CAD models on the 3D scene from all markup lines:

1. Right-click anywhere on the 3D scene.
2. Click Remove All Markup Lines.





All markup lines are removed from the 3D scene and cannot be restored.





## Manual Geometry Transformation

A 3D CAD assembly is a set of viewable SCS and SCZ files in the Dynamic and Streaming Viewers respectively, an end user can manually transform the geometry of one or more SCS and SCZ files from the assembly right in these viewers. As such a file can be a subassembly or leaf part, the user can move and rotate a given subassembly or part on the 3D scene along:

- The X, Y, and Z axes in the coordinate system of this given subassembly or part.
- A surface or edge of another part.

The end user can move and rotate a single subassembly or part with multiple geometry transformation sessions to achieve its necessary position and orientation. Multiple subassemblies and parts can be moved and rotated on the 3D scene.

The user can also add markup lines to the moved and rotated subassemblies and parts as well as move and rotate the ones with existing markup lines. While moving and rotating, the existing markup lines preserve their connections. For more details on markup lines, see the Markup Lines in Dynamic Viewer and Streaming Viewer section.

The Dynamic or Streaming Viewers does not embed manual geometry transformations into 3D CAD model native and viewable files.

A 3D CAD model view with manual geometry transformations can be saved as outlined in the Saved Views section.

To quite the manual geometry transformation of a subassembly or part, either:

- Click on the 3D canvas outside a 3D CAD model.
- Press Escape on the keyboard.

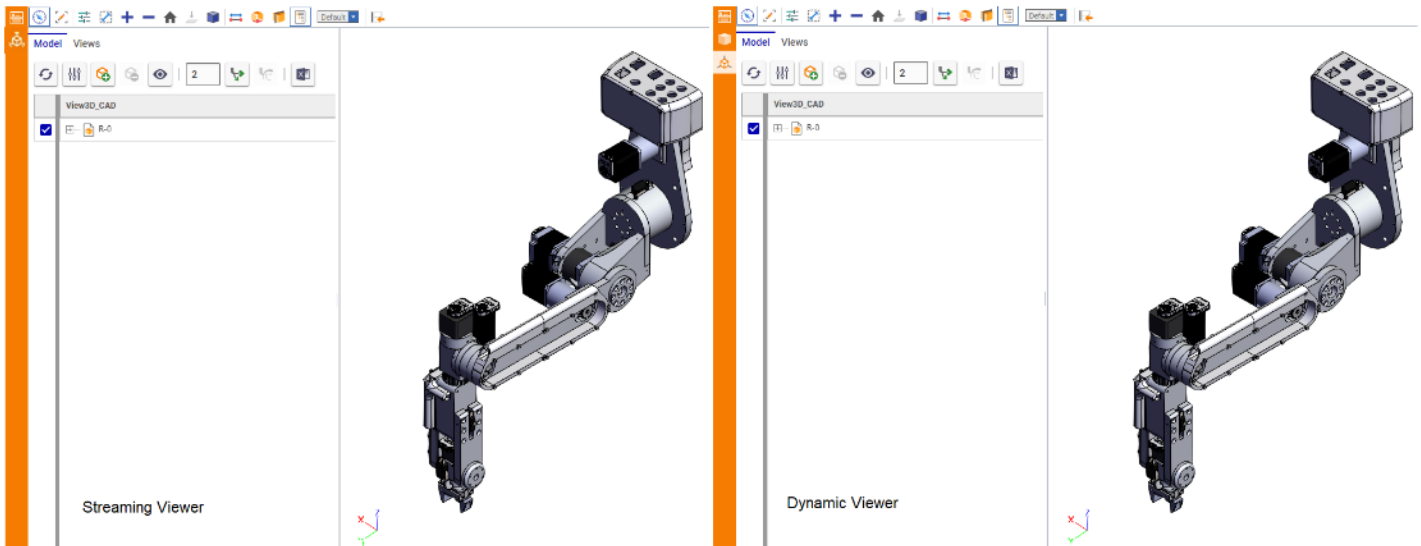


## Model Browser in Dynamic and Streaming Viewers

The **Model Browser** button on the **Standard Viewing** toolbar provides access to the **Model Browser** section.

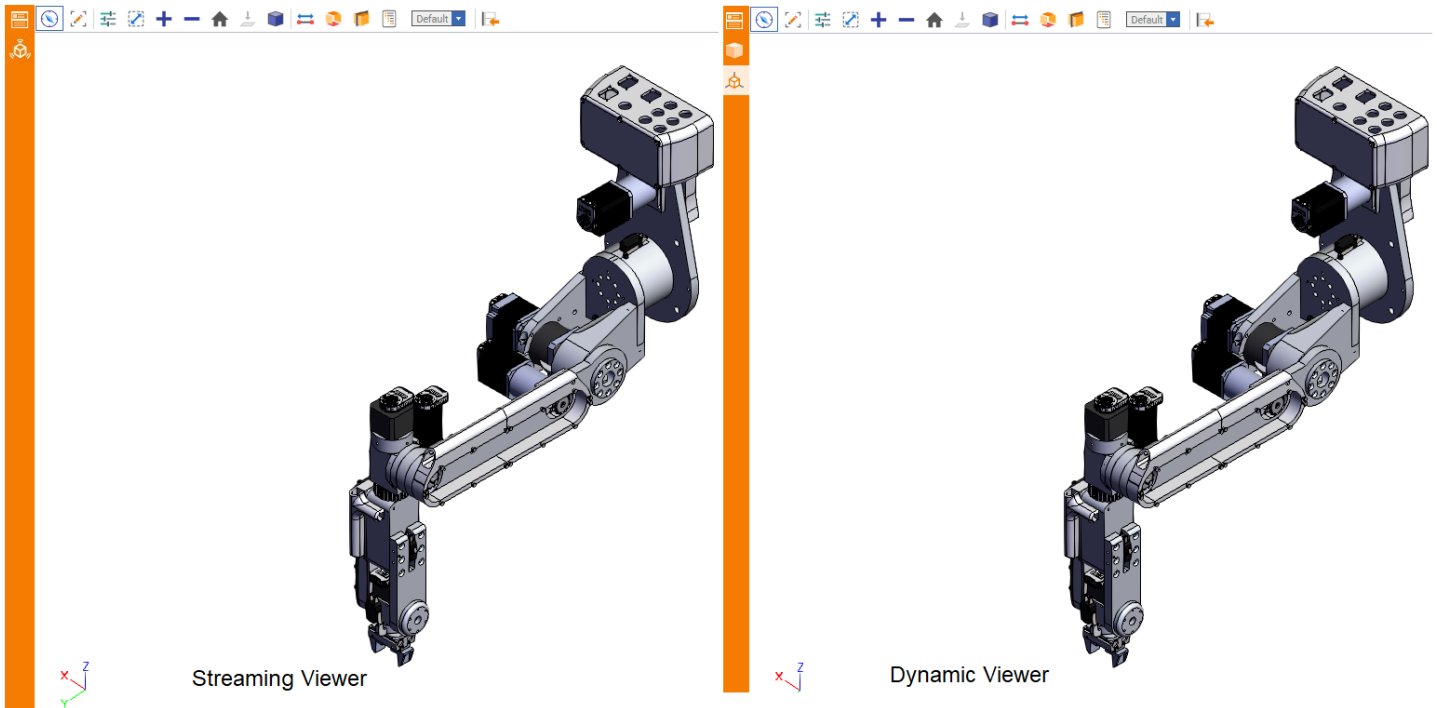


By default, the **Model Browser** section is shown at the first opening of the both Dynamic as well as the Streaming Viewer for a given CAD Document in the current logging session.



Clicking the **Model Browser** button shows or hides the **Model Browser** section right after the left sidebar.





The **Model Browser** section has two tabs:

- **Model:** a Tree Grid View (TGV) showing a tree of CAD Documents included in a CAD assembly representing a given 3D CAD model with capabilities to:
  - Navigate through subassemblies and parts in a given assembly.
  - Quickly select a necessary subassembly or part.
  - Configure which CAD Document versions should be loaded into the Dynamic or in the Streaming Viewers through Structure Resolution Parameters.
  - Create and manage digital mockups.
- **Views:** a list of model views with capabilities to:
  - Create, delete, and restore Saved Views.
  - Switch quickly between viewing positions available for a given 3D CAD model.

## Saved Views

The Dynamic Viewer and the Streaming Viewers both provides end users with the Saved Views feature for restoring 3D scene views of one or more 3D CAD models.

The end users can create custom Saved Views to preserve the current View state as discussed in the *Creating Saved Views* section. When creating a Saved View for a 3D scene view, the following functions applied to this 3D scene view are saved with the Saved View: the selected Dynamic View Definition, applied Parameter Values, selected View Mode, added parts and assemblies, and the current camera position.

Some view parameters cannot be saved and restored; for example, the display style, exploded increment, measurements, and cutting planes.

At any time in the future, end users can apply an existing Saved View to a 3D scene to restore their digital mockup or favorite viewing position as discussed in the Restoring Saved Views section. When restoring a Saved View for a 3D scene view, the abovementioned saved functions are re-applied to the scene.

However, a restored 3D scene view is not guaranteed to match the original 3D scene view because a Saved View includes only input information for a Query Definition which execution may return a different result set.

Saved Views given to a CAD Document will remain associated with all versions of this CAD Document.

End users can delete unnecessary Saved Views as discussed in the 6.6.3 Deleting Saved Views section.

## Multi-Selection

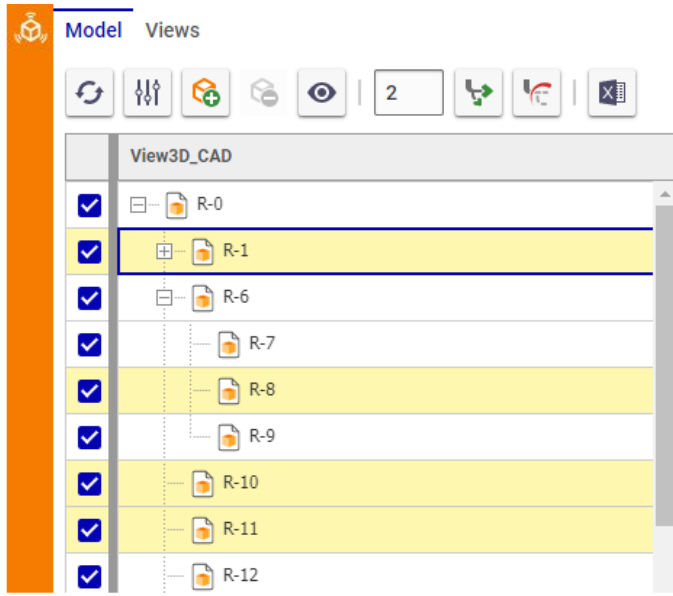
The Dynamic and Streaming Viewers also support multi-selection: one or more subassemblies and parts can be selected during viewing.

There are two ways to select multiple subassemblies and parts:

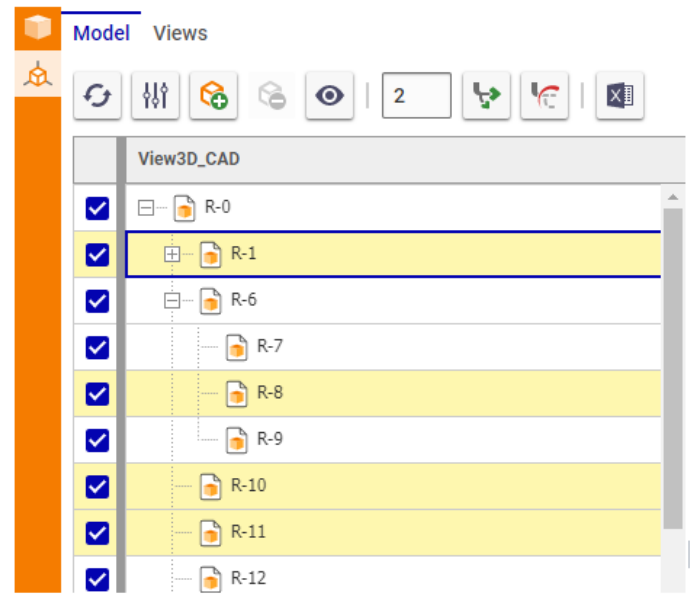
- Click each object on the 3D scene to be selected while pressing the Ctrl key on the keyboard.



- Click each **CAD Document** in the Models TGV grid to be selected while pressing the Ctrl key on the keyboard.



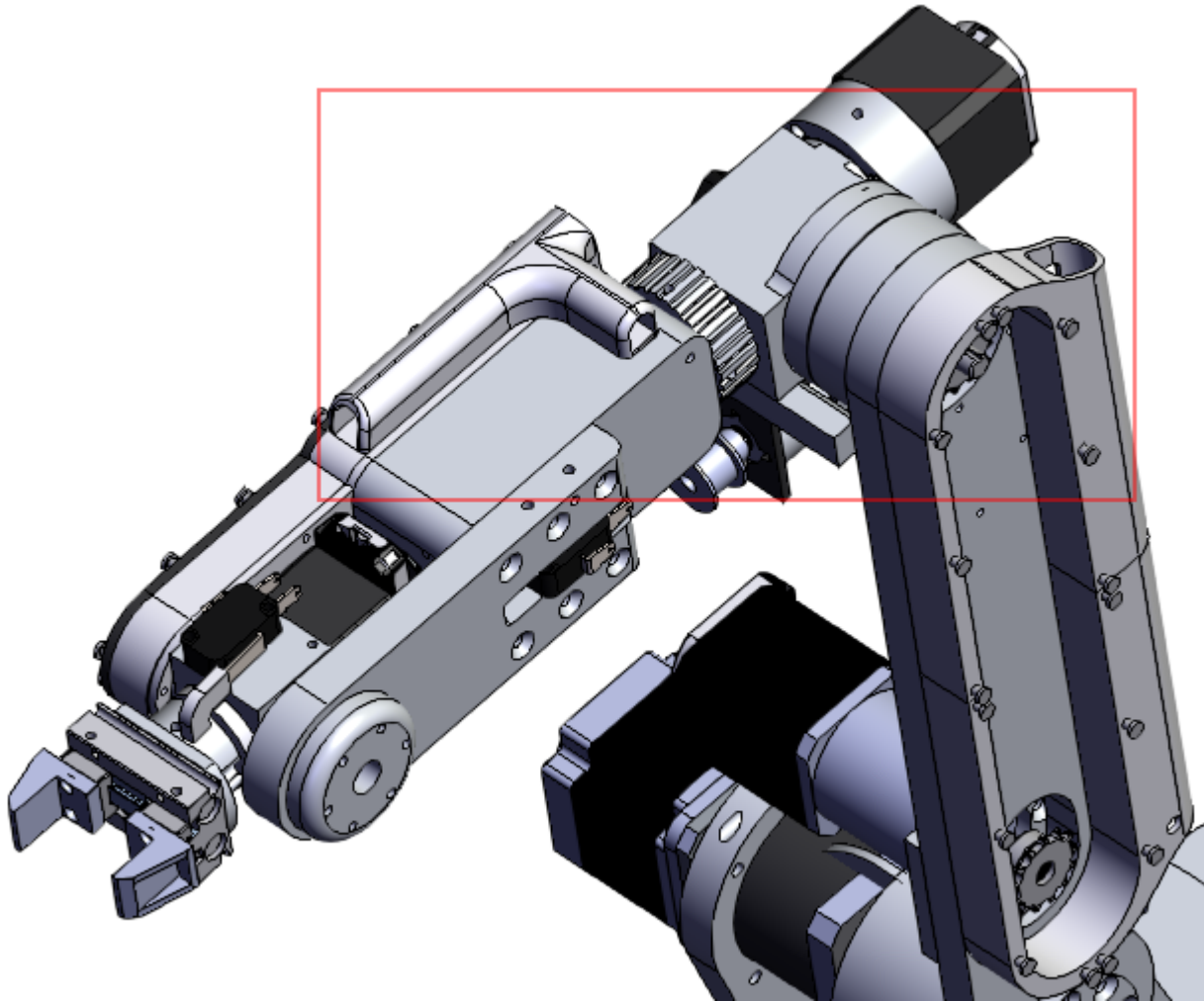
Streaming Viewer



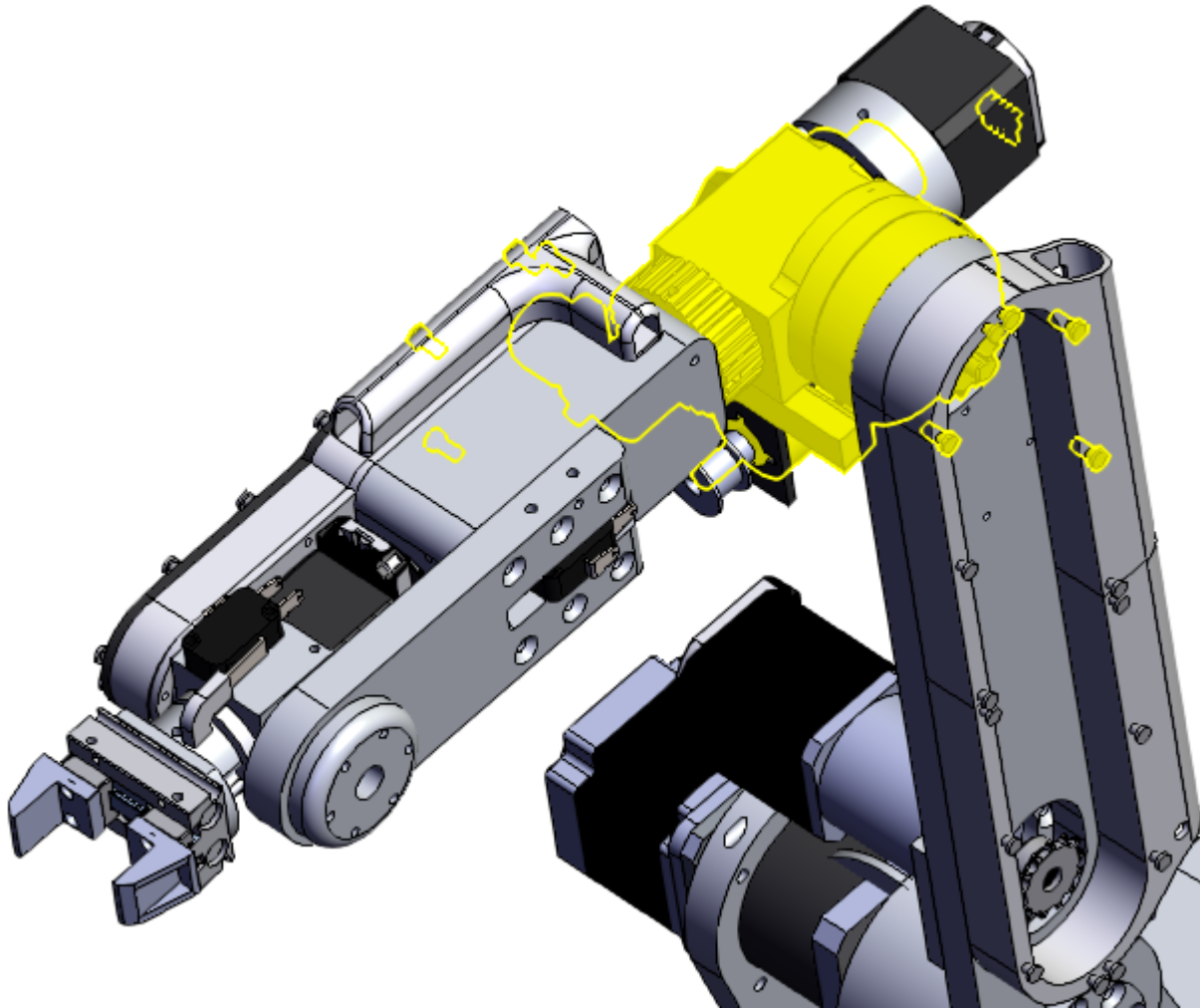
Dynamic Viewer

- Click and drag the cursor on the 3D scene while pressing the Ctrl key on the keyboard to select multiple objects in an arbitrary area. Incremental additions to selection are possible while pressing Ctrl. Objects selected depends on a dragging direction:
- If dragging from left to right, the selection box is red.

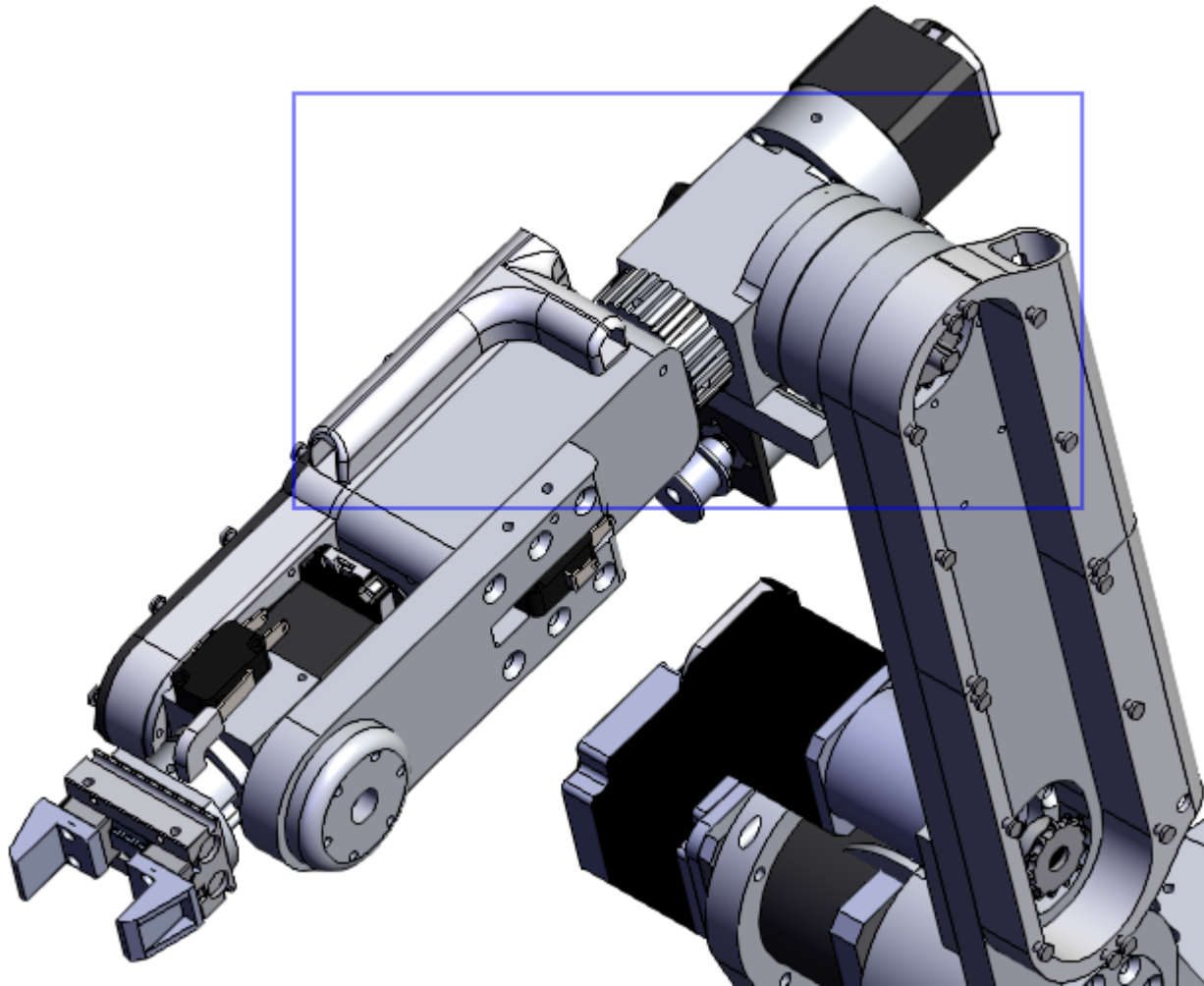




The dragging right will select objects which are completely enclosed in the red selection box area.



- If dragging from right to left, the selection box is blue.



The dragging left will select objects which are partially enclosed in the blue selection box area.

